DOCUMENT RESUME

ED 255 199 IR 011 571

AUTHOR Huber, Leonard; And Others
TITLE Project Logic Handbook: Computer Literacy through BASIC.
INSTITUTION Hampshire Educational Collaborative, South Hadley, MA.
PUB DATE [83]
NOTE 141p.
PUB TYPE Guides - Classroom Use - Guides (For Teachers) (052) -- Computer Programs (101)

EDRS PRICE MF01/PC06 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; Computer Graphics; *Computer Literacy; *Computer Science Education; Curriculum Guides; Educational Objectives; Elementary Secondary Education; Learning Activities; *Microcomputers; *Programing; Programing Languages
IDENTIFIERS *BASIC Programing Language

ABSTRACT

This handbook for teachers offers guidance on introducing computer literacy into elementary and secondary classrooms. It includes a list of computer concepts exemplified by each step in learning to write programs in BASIC Programming Language and the objectives for the elementary and secondary activities; suggestions for using computers in classrooms; and learning activities for elementary and secondary students. For elementary students, activities deal with PRINT and GOTO statements, computer arithmetic, variables and values, INPUT statements, IF-THEN statements, FOR-NEXT loops, string manipulation, random numbers, arrays, nested loops, low resolution graphics, flowcharts, and activities without the computer. Secondary activities cover PRINT statments; GOTO, CTRL C, comma, and semicolon; INPUT command, variables, ON-GOTO; FOR-NEXT; string manipulation; concepts of probability; writing a program in the form of a multiple choice quiz using several commands; READ-DATA; arrays; programming to find roots of a polynomial; LEFT, RIGHT, MID$; word scramble program (concatenation); text files; reaching objectives by using files; writing a library program using text files; text windows; writing programs using a counter; quadratic equations; graphics; and programs to explore the world of low resolution graphics. (LMS)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

X This document has been reproduced as received from the person or organization originating it.

[ ] Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

PROJECT LOGIC HANDBOOK: Computer Literacy Through BASIC

HAMPSHIRE EDUCATIONAL COLLABORATIVE
South Hadley, MA 01075

LEONARD HUBER, PH.D.
Project Director

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Leonard Huber

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Preface

This handbook is the basis for introducing computer literacy into elementary and secondary classrooms. It includes the following components: (1) Teacher Training Manual; (2) Curriculum; (3) Elementary BASIC Programming Activities; (4) BASIC Programming Activities for Secondary Students.

The first step in implementing a computer literacy curriculum is for teachers to become computer literate. The Teacher Training Manual ("Helping Teachers Become Computer Literate: A BASIC Approach") lays the groundwork for this. In going through the manual, it is helpful to have someone on hand who is familiar with BASIC. However, outside assistance is not essential, if teachers are willing to spend a few minutes scratching their heads in bewilderment, once in a while. The interaction among a small group of teachers can usually solve problems far more difficult than learning BASIC.

After teachers become familiar with BASIC, they can introduce it to their students. There are some fundamentals of this process that are covered in the curriculum section. The curriculum section also lists the computer concepts exemplified by each step in learning to write programs.

The two activity manuals are designed for teaching elementary and secondary students. However, teachers at any level should feel free to borrow from either book. There are good activities in the secondary manual for advanced elementary students, and there are good activities in the elementary manual for beginning secondary students. There are also a few topics that are covered in only oe manual or the other (e.g., flowcharts, text files).

Finally, a word about BASIC. Another computer language, Logo, has received a lot more attention lately. However, teachers should know that BASIC has several advantages: (1) it is far more widespread, and will be for a long time because it requires very little computer memory; (2) it makes certain computer concepts easy to understand, such as INPUT, ARRAYS, and TEXT FILES. These concepts are hard, or impossible, to explore in Logo.

PROJECT LOGIC HANDBOOK:
Computer Literacy Through BASIC

CONTENTS

Part I	Curriculum
Part II	Fundamentals of Using Computers in Classrooms
Part III	Elementary BASIC Programming Activities
Part IV	BASIC Programming Activities for Secondary Students

Project Logic Curriculum

Project Logic introduced computers to students at the elementary and secondary levels. For both groups, computer literacy objectives were more or less similar. However, the secondary students tended to tackle more difficult programming projects as they studied the various concepts. This is particularly clear regarding the topic of TEXT FILES. These files are a key part of most commercial software; they are covered extensively in the secondary manual and not at all in the elementary manual. Nevertheless, the principal they exemplifiy--orderly data storage--is amply considered in conjunction with other elementary activities (READ-DATA and ARRAYS).

Because of the differing needs and capacities of elementary and secondary students, we have separate activity manuals for the two levels. The following table lists Project Logic's curriculum objectives. Next to each objective are the titles of the activities designed to achieve it, divided into two columns. One column is for elementary and one for secondary. Key vocabulary terms are typed in capital letters.

Objective: Students will understand that...	Elementary Activity	Secondary Activity
1. PRINTing on the screen is a form of computer OUTPUT.	PRINT and GOTO statements	PRINT statements
2. A computer can repeat a series of steps endlessly.	PRINT and GOTO statements	GOTO, CTRL-C, comma & semicolon
3. One of the things that computers do best is COMPUTE.	COMPUTER ARITHEMETIC	Concepts of Probability *** Programming to find roots of a polynomial *** Algebra II
4. A VARIABLE can be thought of as a place in the computer where a number or a word is stored.	VARIABLES and VALUES	Variables
5. The thing that is stored in a VARIABLE is its VALUE.	VARIABLES and VALUES	Variables
6. VARIABLES which contain letters or special characters are called STRING variables.	VARIABLES	Variables
7. The INPUT to a computer program is one factor in determining its OUTPUT.	INPUT Statements	Input Command

8. A good computer program PROMPTS the user for input.	INPUT Statements	Input Command
9. Computer programs are flexible because they branch.	IF-THEN Statements	ON-GOTO
10. The IF-THEN statement controls BRANCHing.	IF-THEN Statements	Input Commands
11. Printed OUTPUT can be produced in a variety of formats.	More on PRINTing	PRINT Statements
12. Programming languages contain more than one way to control loops.	FOR-NEXT loops	FOR-NEXT
13. STRING manipulation is a process parallel to performing arithmetic on variables.	STRING Manipulation	LEFT, RIGHT, MID *** Word Scramble Program
14. Computers are helpful because they can SIMULATE random processes.	RANDOM NUMBERS	Concepts of Probability

15. A good computer program stores data in an orderly fashion.	READ-DATA *** ARRAYS	READ-DATA *** ARRAYS *** Text Files, MON *** Text Files *** Reaching Objectives by Using Files *** Writing a Library Program Using Text Files
16. In a good computer program, the order of operations is clear.	NESTED LOOPS	ON-GOTO
17. COMPUTER GRAPHICS define pictures in mathematical terms.	LOW RESOLUTION GRAPHICS	GRAPHICS *** Programs to Explore the World of Low Resolution Graphics
18. Computer programs should be planned in advance.	FLOWCHARTS	
19. The steps of a computer program can be "walked through" without a computer.	Activities without the Computer	
20. The computer has an OPERATING SYSTEM which is independent of any particular programming language. However, the operating system can be controlled by BASIC commands.		CHR$

Fundamentals of Using Computers in Classrooms

Whenever computers are brought into classrooms they quickly come to serve a multitude of purposes. Even if computer literacy is carefully taught according to a detailed curriculum, new activities quickly absorb computer resources. These activities include word-processing, computer-assisted instruction, games, and administrative functions.

Computers have diverse applications in the real world, and it's good to show children that they have many uses. Fortunately, despite the myriad alternative activities, there are many constants in using computers educationally. They cut across hardware and software decisions. Here are some you should try to keep in mind.

1. Use computers to help students achieve educational objectives you believe in.

A computer programming activity can teach language arts or math skills or even social studies. Computer science need not be introduced as a new curriculum area.

2. Plan and try out lessons in advance.

Students need leadership with the computer as with any other classroom activity. It is true that they can amuse themselves for quite a while without any instruction, but such time can be worthless.

3. Use paper and pencil in conjunction with the computer.

Good programmers plan in advance with paper and pencil, and make notes as they go along. Computers are probably a scarce resource in your school; children should be primed to use them when they are available. A computer journal is a good idea for budding programmers. On the other hand, don't let clumsy handwriting block computer progress. At the younger grades, it's the teacher who should make charts to keep track of new learning.

4. Look at mistakes as valuable feedback.

You will often hear students complain "How can I fix that?" or "How can I get rid of that?" Encourage the complainer to relax and understand the meaning of the result which is perturbing her/him. Build this attitude into a habit.

5. Treat the computer and its associated materials with care.

For example, avoid turning the computer off and on as a way of interrupting programs. There is usually a keypress which will do the same job without wearing on the power switch. Similarly,

don't leave disks and cassettes lying around. If your computer has a plug-transfonner, be sure to unplug it when you are not using the computer. It keeps on working even when the computer is turned off.

6. Be prepared to save student work on disk or cassette.

The students deserve to work cumulatively from session to session, rather than starting from scratch each time. This is particularly true when they are working on lengthy projects. Saving work electronically cuts down on typing, and typing is a major bottleneck for children using computers.

7. Get involved with computers.

Don't let an aide or parent volunteer become your resident expert. If you do, you will soon feel that the students know more than you, and you will feel less and less confident about using the computer with them. Consider using the computer recreationally. There are very few hobbies without computer applications. For example, programs for the Apple II are useful in gardening and weaving.

Elementary BASIC Programming Activities

by
Leonard Huber &
Martin Urbel

Project Logic

Hampshire Educational Collaborative
South Hadley, MA 01075

INTRODUCTION

This booklet is designed to be an aid to persons teaching beginning BASIC. It is the result of our experience teaching BASIC to elementary school children, as part of Project Logic. This project was supported by Title IV-C in the Commonwealth of Massachusetts, and was one of the pioneer computer education projects in New England. We extend our sincere thanks to the officers of the State Department of Education who made this work possible—Al Trocchi and Jack Reynolds. We also thank participating staff from the Hampshire Educational Collaboartive and from the participating School Districts—Hadley and Williamsburg.

Although the book is targeted for the elementary school teacher, we think that it also has value for the teacher of adult novices. We assume that the teacher already has some familiarity with BASIC on the Apple (i.e., Applesoft BASIC).

Each activity has RULES and EXERCISES. The RULES are intended to serve as a refresher and may not always be comprehensive. You can get more details by consulting our manual "Helping Teachers Become Computer Literate: a BASIC Approach", or any of the books in the bibliography. The first EXCERCISES in each activity are very simple and can be used with children as young as 6. They increase in complexity. Emphasis is on use of words rather than numbers on the assumption that children are more familiar and comfortable with words. For more advanced exercises, consult our manual "BASIC Programming Activities for Secondary Students."

The EXCERCISES should be assigned at the discretion of the teacher, taking into account student interest, programming ability, goals, and time available. Many of the exercises provide a structure from which you can generate more activities. Some of the assignments are programs of 10 or 20 or even more lines. They can be assigned to the students to write, or presented to run, study, and learn the principals involved. These programs are contained on the accompanying diskette, so you can try them out without having to key them into the computer. Students should be encouraged to write their own original programs whenever possible.

As for the order of activities, the lessons on PRINT and GOTO statements, COMPUTER ARITHMETIC, STORING VALUES, and INPUT STATEMENTS should be given first, in order. The others can, for the most part, be used at the discretion of the teacher. The lesson on LOW RESOLUTION GRAPHICS can be presented at any time. It is independent of the other topics. The exercise in the graphics lesson on writing a program to display a letter of the alphabet can serve as a good introduction to writing programs of substance. Prerequisites for other lessons or for more advanced excercises within a lesson are indicated.

The diskette containing the longer programs also contains the text of the lessons in both text and binary form. The files were written with the PIE Writer word processing system, but can be edited with most other Apple-compatible word processors. The

file names are preceded by "B-" for binary files and "T-" for text files. We encourage you to improve the activities in any way you see fit.

Len Huber
Martin Urbel
6/1/83

Filename: INTRO

topic:PRINT and GOTO statements

filename:PRINTGOTO

RULES:

*Characters included between quotation marks (") will be printed exactly as shown.

*The characters between quotation marks are called a STRING.

*If the closing quotation mark is followed by a comma, the next string will be printed on the same line, at the next tab position. The result is to create columns.

*If the closing quotation mark is followed by a semicolon, the next string will be printed on the same line, immediately following. The effect is to cause strings to run into each other.

*The question mark (?) may be used as an abbreviation for PRINT. When the program is LISTed, the computer substitutes PRINT for ?.

*Control of the program branches to the line whose number follows the GOTO.

*GOTO statements can create "infinite loops", i.e., loops which have no end. Interrupt such loops by holding the control key down and typing C. (This is abbreviated as ctrl-C.)

EXCERCISES:

*Enter each of the following instructions and compare the results. (Substitute actual names where appropriate.) Pay close attention to the use of punctuation and spaces.

```
PRINT "your first name";"your last name"
PRINT "your first name ";"your last name"
? "your first name","your last name"
```

*Now, try it with line numbers. Remember, begin by typing NEW. Then, enter the following line, LIST it, and RUN it.

```
10 ? "your name"
```

*Incorporate the GOTO statement by adding...

```
20 GOTO 10
```

List the program and run it. Remember, CTRL-C will interrupt a program (and break an infinite loop).

*Retype line 10 with a semicolon at the end. What is the effect?

*Retype line 10 with a comma at the end. What is the effect?

*Create other patterns on the screen using PRINT and GOTO instructions.

*Enter the following programs and RUN them.

```
10 ? 2+2
20 PRINT 5/10

10 PRINT "2+2"
20 PRINT "5/10"

10 X=5
20 PRINT "X=";X

10 X=10
20 PRINT "X=",X
```

topics: COMPUTER ARITHMETIC, ORDER OF OPERATIONS

filename: ARITHMETIC

RULES:

*The asterisk (*) is used for multiplication; the slash (/) is used for division; and SHIFT-N (^) is used for exponentiation.

*Operations within parentheses are done first. Next comes exponentiation. Then, multiplication and division are performed from left to right. Finally come addition and subtraction.

EXCERCISES:

Predict the outcome of each of the following commands and compare with the result given by the computer. Create your own equations for students to solve, geared to their arithmetic skills.

```
PRINT 2+3-4

PRINT 6*4

PRINT 6/4

PRINT 4+2.5*3,7-11

PRINT 3^2

PRINT 3+6/3

PRINT (3+6)/3

PRINT 3*2+2*5

PRINT 16/2*4

PRINT 4*2^2+1/2*4+3

PRINT 4*2^(2+1)*4+3
```

opic:VARIABLES and VALUES

filename:VARIABLES

RULES:

*A variable is a pigeonhole where a number or a string of letters is stored. The thing that is stored is called the value of the variable.

*A variable can have almost any name as long as it begins with a letter.

*Applesoft BASIC allows for STRING variables and two types of numeric variables, REAL and INTEGER. We use STRING and REAL variables.

*A STRING is a sequence of characters enclosed in quotation marks, e.g. "computer", "5+4". The name of a string variable must end with a dollar sign ($), e.g., N$, NAME$

*A REAL variable contains a numeric value which may be be expressed as a whole number or decimal. (Very large or very small numbers may be expressed by the computer in terms of an exponent. If a number appears with the letter "e" in it, consult the reference manual—or avoid such numbers.) REAL variable names should not include any punctuation marks. Examples of real variable names are N, NUMBER, X1, X2.

*Only the first two characters of a variable name are recognized. The computer considers NA$, NAME$, and NAT$ to be the same name.

*Variables can contain only one value at a time. A variable has a zero stored as its value until it is assigned another.

*An equal sign sets the value of of the variable to its left to be equal to the number, string, or value of the variable, on its right.

EXCERCISES:

*Enter the following commands. Try to guess what result they will produce.

```
X=5
?X

X=X+3
?X

X$=DAVE
?X$

X$="DAVE"
?X$
```

*Predict the output of the the following programs. Then run them and compare the results.

```
10 X=5
20 Y=10
30 Z=X+Y
40 PRINT X,Y,Z
50 X=X+1
60 Z=Y-X
70 PRINT X,Y,Z

10 A$="THE SUM OF A AND B IS "
20 A = 5
30 B = 10
40 PRINT A$;A+B
```

*Create new programs which do arithmetic and print the result. Especially focus on problems that would be difficult to solve with paper and pencil.

*See lessons on INPUT and GRAPHICS for more practice.

topic:INPUT STATEMENTS

filename:INPUT

prerequisites: PRINT, VARIABLES

RULES:

*An INPUT statement causes the computer to print a question mark and wait for the user to type in a response. The value of the response is stored in a variable. For example, the response to INPUT A is stored in the variable A. (It must be a number.) The response to INPUT A$ must be a STRING.

*The computer can be programmed to give a "prompt" before accepting input. For example, a statement like...

```
INPUT "WHAT IS YOUR NAME?";N$
```

causes the string in quotes to be printed without a question mark being added by the computer. The response is stored in the variable named N$. (Of course, any string can replace the string used in the example, and any variable name can replace N$.)

EXCERCISES:

*Run the following one line program. Type in your name when the computer prints "?".

```
10 INPUT X$
```

Now ask the computer, PRINT X$. Do you understand the result? Repeat the process several times, with different names. Observe how each name is stored in the variable X$, and replaces the previous value of X$.

*Add the following lines to your program:

```
20 INPUT Y$
30 PRINT X$+Y$
```

Run the program, entering a name in response to each question mark. Do you understand the result? Now, run the program, entering a number in response to each question mark.

Rewrite the program, without the dollar signs. Try it with names and numbers and note the differences.

*The following program uses a simple form of prompts (i.e., questions which tell the user what kind of information to type in). The program uses a separate PRINT statement for each prompt.

```
10 PRINT "WHAT IS YOUR NAME?"
20 INPUT N$
30 PRINT "WHAT IS YOUR FAVORITE FOOD?"
40 INPUT F$
50 PRINT N$;"'S FAVORITE FOOD IS ";F$
```

```
]LOAD MAD-LIBS
]LIST

5  PRINT "LETS PLAY MAD-LIBS!"
6  PRINT "I'LL ASK YOU FOR SOME WORDS AND THEN SUBSTITUTE THEM FOR WORDS I
     N A FAMOUS SPEECH."
10  PRINT "PLEASE TYPE IN A PLURAL NOUN"
20  INPUT N1$
30  PRINT "PLEASE TYPE IN ANOTHER PLURAL NOUN."
40  INPUT N2$
50  INPUT "NOW, AN ADJECTIVE: ";A1$
60  INPUT "A SINGULAR NOUN: ";N3$
70  INPUT "A PLURAL NOUN: ";N4$
80  INPUT "AND FINALLY, AN ADJECTIVE: ";A2$
85  PRINT : PRINT
90  PRINT "FOURSCORE AND SEVEN ";N1$;" AGO, OUR "N2$;" BROUGHT FORTH UPON
     THIS CONTINENT A ";
100  PRINT A1$;" ";N3$;" DEDICATED TO THE PROPOSITION THAT ALL ";N4$;" ARE
      CREATED ";A2$

]
```

The following program uses the prompt in the INPUT statement. Notice the difference in the way the computer asks its prompting question.

```
10 INPUT "ENTER A NUMBER ";N1
20 INPUT "ENTER ANOTHER NUMBER ";N2
30 PRINT "THE SUM OF ";N1;" AND "N2;" IS ";N1+N2
```

Now, create some programs off your own which request information and then print some interesting results. For example, write a program which asks a user their age and then tells them how old they will be in the year 2000. Or, try one based on compound interest—if you invest a sum of money at 8% interest, how much will you have in 1999?

*LOAD the program MAD-LIBS from the diskette and RUN it. LIST the program and follow its steps. Create your own version of Mad-Libs using a different text.

topic: LOW-RESOLUTION GRAPHICS

filename:GRAPHICS

RULES:

*To get into the graphics mode type GR. The command GR always clears the screen.

*The screen is divided into 40 horizontal rows numbered 0 to 39 and 40 vertical columns, also numbered 0 to 39. These rows and columns intersect to form 1600 "bricks" numbered 0,0 to 39,39. (See attached grid.)

*In the graphics mode, the bottom three lines of the screen are reserved for text.

*There are 16 colors, numbered 0 to 15. To choose a color, type "COLOR = " followed by a number. For example, to draw in green, type...

```
COLOR = 12
```

* The command PLOT, followed by a pair of numbers, lights up the small "brick" on the screen. The location of the brick is specified by the number pair. For example, the command....

```
PLOT 20,20
```

lights up the brick at the center of the screen.

*The command HLIN draws a horizontal line on the screen. The beginning point and end point of the line are specified by a pair of numbers. The line will occupy a row specified by a third number. For example, the command...

```
HLIN 0,20 AT 30
```

draws a line which begins at the left edge of the screen (0) and extends to the middle of the screen (20). The line occupies a row (30) towards the bottom of the screen.

*VLIN, followed by three numbers, draws a vertical line. For example,

```
VLIN 0,39 AT 39
```

draws a vertical line down the right edge of the screen.

*To correct a mistake change the COLOR to the color of the background (0 if black) and retype the mistake.

EXCERCISES:

*Load the program GRCOLORS and run it. (This program shows the colors available and their corresponding numbers.)

*Load the program DEMO24 and run it. The program demonstrates the grid and provides practice using

coordinates for the PLOT command.

*Make photocopies of the attached grid and give one or more to each student. Tell students that to use the grid, they should lightly fill in blocks so that numbers are still visible.

*Use the grid to figure out how to command the computer to light up the corner bricks and the center brick.

*Use the PLOT command to Draw a horizontal line 5 blocks across. Draw a vertical line on the grid, and then PLOT it.

*Now use the HLIN and VLIN commands to draw lines next to those above. Draw them in a new color.

*Make several figures someplace else on the grid. Select a new color and plot them on the screen, using PLOT, HLIN, and VLIN.

*Before students get too far in making elaborate designs, ask them if they want to save their creations. If yes, start a program. For example...

```
10 GR
20 COLOR = 12
30 VLIN 12,30 AT 20
etc.
```

This exercise shows children the advantage of writing programs rather than always operating in the immediate mode: if one line isn't quite right, it simply needs to be retyped.

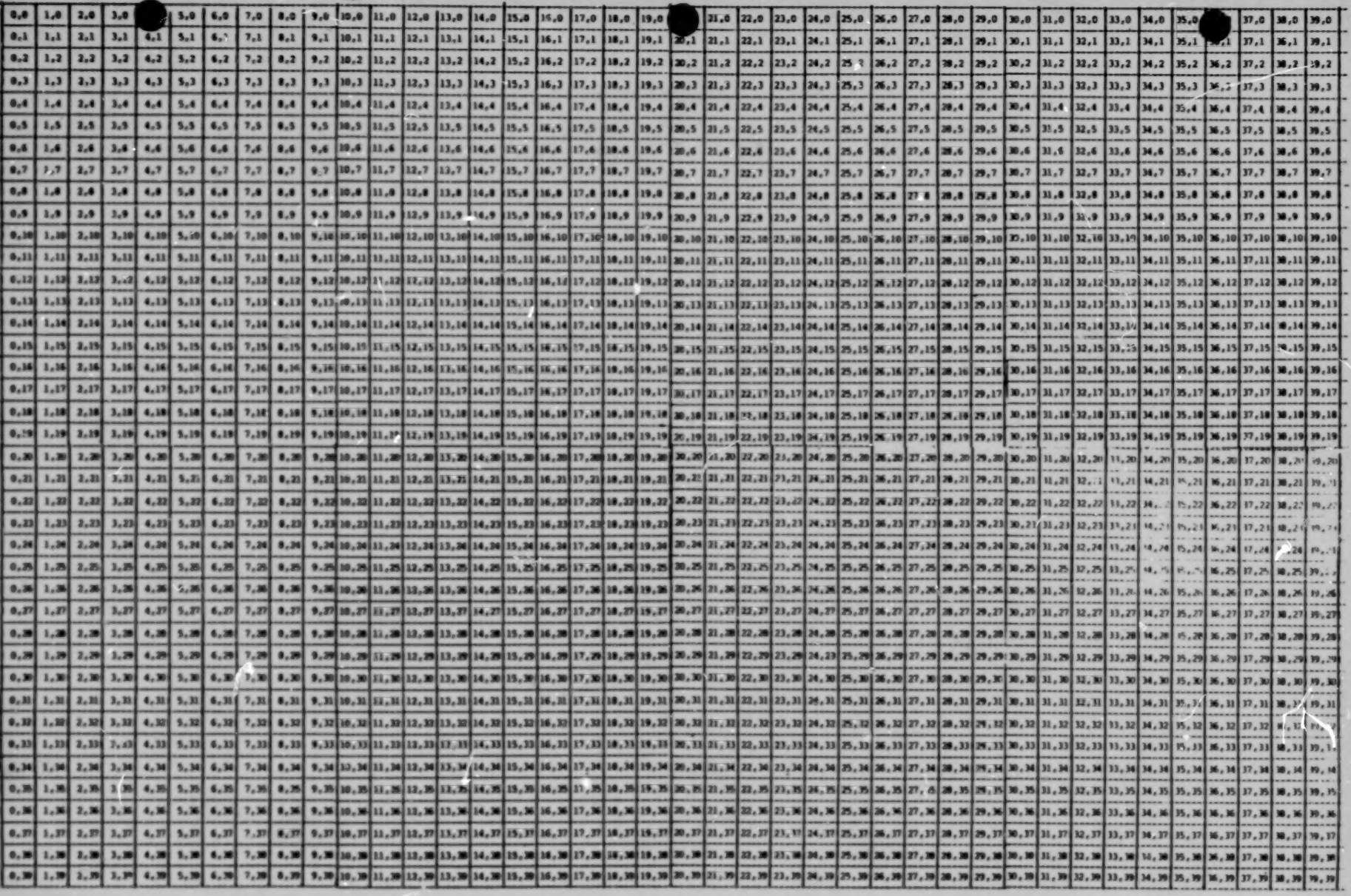
*Draw your first initial and put it on the screen.

*Create a program which will draw your initial. (See program LETTERA or LETTERN for sample solutions.)

*prerequisites: INPUT, VARIABLES. Change the program from the previous excercise so that it will accept grid coordinates as input, and display the letter in that location. (See MOVEA and MOVEN for sample solutions.)

*prerequisites: VARIABLES, IF, GOTO. Write a program which draws a diagonal line from the top left corner of the screen to the bottom right (or have students run Program: DIAG1). Write a program which draws a line between the other two corners (Program DIAG2).

BEST COPY AVAILABLE



```
10 D$="----------"
20 INPUT X,Y
30 PRINT D$
40 FOR J = Y TO Y + 19
50 FOR I = X TO X +9
60 PRINT "1";
70 IF I<10 THEN PRINT " ";
80 PRINT I;",";J;
90 IF J<10 THEN PRINT " ";
100 IF I = X + 9 THEN PRINT "1":PRINT D$
110 NEXT I
120 NEXT J
```

For use with APPLE II
low resolution graphics.

Project Logic

HAMPSHIRE EDUCATIONAL
COLLABORATIVE

BEST COPY AVAILABLE

```
]LOAD GRCOLORS
]LIST

5  REM :PROGRAM NAME: GRCOLORS
10  GR
20  FOR I = 0 TO 31
30  COLOR= I / 2
40  VLIN 0,39 AT I
50  NEXT I
60  FOR I = 0 TO 14 STEP 2: PRINT  TAB( I * 2 + 1);I;: NEXT I
70  PRINT
80  FOR I = 0 TO 15 STEP 2: PRINT  TAB( I * 2 + 1);I;: NEXT I
90  PRINT : PRINT "STANDARD APPLE COLOR BARS";

]
```

```
]LOAD DEMO24
]LIST

5  REM  PROGRAM NAME: DEMO24
7  REM  THIS PROGRAM APPEARS ON PAGE 24 OF THE APPLESOFT II "BASIC PROGRAM
     MING REFERENCE MANUAL"
10  GR
20  HOME
30  COLOR= 1: PLOT 0,0
40  LIST 30: GOSUB 1000
50  COLOR= 2: PLOT 39,0
60  HOME : LIST 50: GOSUB 1000
70  COLOR= 12: PLOT 0,39
80  HOME : LIST 70: GOSUB 1000
90  COLOR= 9: PLOT 39,39
100  HOME : LIST 90: GOSUB 1000
110  COLOR= 13: PLOT 19,19
120  HOME : LIST 110: GOSUB 1000
130  HOME : PRINT "PLOT YOUR OWN POINTS"
140  PRINT "REMEMBER, X & Y MUST BE  =0 &  =39"
150  INPUT "ENTER X,Y: ";X,Y
160  COLOR= 8: PLOT X,Y
170  PRINT "TYPE CTRL-C AND PRESS RETURN TO STOP"
180  GOTO 150
1000  PRINT "***HIT ANY KEY TO CONTINUE***";: GET A$: RETURN

]
```

```
]LOAD LETTERA
]LIST

10  REM : THIS GRAPHICS PROGRAM PRODUCES THE LETTER "A"
20  GR
30  COLOR= 12
40  PLOT 9,0
50  VLIN 1,5 AT 8
60  VLIN 1,5 AT 10
70  PLOT 9,3

]
```

```
]LOAD LETTERN
]LIST

10  REM : PROGRAM NAME:LETTERN
20  REM : THIS GRAPHICS PROGRAM PRODUCES THE LETTER "N"
30  GR
35  COLOR= 12
40  VLIN 0,5 AT 25
50  PLOT 26,1
55  PLOT 27,2
60  PLOT 28,3
70  VLIN 0,5 AT 29

]
```

```
]LOAD MOVEA
]LIST

5  REM : PROGRAM NAME MOVEA
10  REM : THIS GRAPHICS PROGRAM PRODUCES THE LETTER A ANYWHERE ON THE SCRE
     EN
20  GR
30  COLOR= 12
40  PRINT "ENTER THE COORDINATS OF THE POSITION WHERE YOU WANT THE TOP LEF
     T BLOCK OF THE LETTER 'A' TO BE PRINTED. "
45  INPUT X,Y
50  VLIN Y,Y + 5 AT X
60  PLOT X + 1,Y
70  VLIN Y,Y + 5 AT X + 2
80  PLOT X + 1,Y + 3
90  PLOT X,Y + 3
100  GOTO 40

]
```

```
]LOAD MOVEN
]LIST

10  REM : PROGRAM MOVEN
20  REM :THIS GRAPHICS PROGRAM PRODUCES THE LETTER 'N' ANYWHERE ON THE GRI
     D.
30  GR
40  INPUT "ENTER THE NUMBER OF THE COLOR YOU WANT. ";C
50  COLOR= C
60  INPUT "ENTER THE COORDINATES OF THE POSITION WHERE YOU WANT THE TOP LE
     FT BLOCK OF THE LETTER 'N' TO BE PRINTED. ";X,Y
70  VLIN Y,Y + 5 AT X
80  PLOT X + 1,Y + 1
90  PLOT X + 2,Y + 2
100  PLOT X + 3,Y + 3
110  VLIN Y,Y + 5 AT X + 4

]
```

```
]LOAD DIAG1
]LIST

5  REM :PROGRAM NAME:DIAG1
10  REM : THIS PROGRAM DRAWS A DIAGONAL LINE FROM THE TOP LEFT PART OF THE
      SCREEN TO THE BOTTOM RIGHT
20  GR
30  COLOR= 12
40 X = 0:Y = 0
50  PLOT X,Y
60 X = X + 1:Y = Y + 1
70  IF X    = 39 GOTO 50

]
```

```
      DIAG2
]LIST

5  REM  :PROGRAM NAME:DIAG2
10  REM : THIS PROGRAM DRAWS A DIAGONAL LINE FROM THE BOTTOM LEFT CORNER O
     F THE SCREEN TO THE TOP RIGHT.
20  GR
30  COLOR= 12
40 X = 0:Y = 39
50  PLOT X,Y
60 X = X + 1:Y = Y - 1
70  IF X    = 39 GOTO 50

]
```

topic: IF-THEN STATEMENTS

filename:IF-THEN

RULES:

*If the condition after IF is true, the statement after THEN is executed. Otherwise, control simply passes on to the next statement.

*The following symbols may be used in the IF statement:

=	equal to
<	less than
>	greater than
<> or ><	not equal to
<=	less than or equal to
>=	greater than or equal to

EXCERCISES:

*prerequisites: PRINT, GOTO, CTRL-C.
Run the following program. Do you understand how it works?

```
10 N=1
20 PRINT N
30 N=N+1
40 GOTO 20
50 END
```

Now replace line 40 as follows.

```
40 IF N<10 THEN GOTO 20
```

*Write a program to count by two's to 100.

*Examine and RUN the program LUCKY NAME. Modify it to use different names, even the names of objects.

*Examine and RUN the program GUESS. Modify it to use colors instead of numbers.

```
]LOAD LUCKY NAME
]LIST

5  REM  PROGRAM NAME :LUCKY NAME
6  REM  MASALSKI PAGE 47
10  PRINT "GUESS WHICH OF THE FOLLOWING IS THE"
20  PRINT "LUCKY NAME: BILL, MARK, JOHN, KATHY"
30  INPUT N$
40  IF N$ = "BILL" THEN  GOTO 70
50  PRINT "SORRY, YOU LOSE."
60  GOTO 80
70  PRINT "RIGHT!, YOU WIN A NEW MICROCOMPUTER!"
80  END

]
```

```
]LOAD GUESS
]

]LIST

5  REM  PROGRAM NAME:GUESS
6  REM  MASALSKI PAGE 50
10  PRINT "GUESS A WHOLE NUMBER BETWEEN 1 AND 10."
20  INPUT N
30  IF N = 7 THEN  GOTO 100
40  PRINT N;" IS NOT THE NUMBER I WANTED."
50  PRINT "DO YOU WANT TO GUESS AGAIN?"
60  PRINT "TYPE YES OR NO."
70  INPUT A$
80  IF A$ = "YES" THEN  GOTO 10
90  GOTO 110
100  PRINT "RIGHT! ";N;" IS THE NUMBER I WANTED!"
110  PRINT "GOODBYE FOR NOW."

]
```

topic: More on PRINTing

filename: TAB

RULES:

*VTAB sets the vertical location of the cursor. For example,

```
VTAB 10
```

moves the cursor to the 10th line on the screen. Any number between 0 and 39 can be used.

*HTAB sets the horizontal location of the cursor. For example,

```
HTAB 20
```

moves the cursor to the 20th position in the line. Any number between 0 and 39 can be used.

*TAB only works in PRINT statements. For example, the command...

```
PRINT TAB(14) "HELLO"
```

starts the word "hello" in the 14th column.

*HOME clears the screen.

*TEXT returns the screen to all-text following a GR instruction.

*The colon (:) is used to separate multiple instructions on one numbered program line.

EXCERCISES:

*prerequisites: GOTO, PRINT, CTRL-C.
Design a program which fills the screen with a particular letter. Then count the letters to determine the dimensions of the text screen. For example...

```
10 ? "X";
20 GOTO 10
```

How many rows and columns are there to the text screen? How many characters can therefore be printed in total. (Answers: 24, 40, 960.)

*prerequistes: FOR-NEXT, GOTO, PRINT, CTRL-C.
Write a program which directly prints numbers to demonstrate the dimensions of the text screen. For example....

```
10 FOR I = 0 TO 9
20 ? I;
30 NEXT I
40 GOTO 10
```

BEST COPY AVAILABLE

*RUN the following program:

```
10 FOR N = 1 TO 960
20 ?"X";
30 NEXT N
40 VTAB 5: HTAB 5
50 PRINT "HELLO"
```

Change the following lines and RUN.

```
40 VTAB 5
50 PRINT TAB (5) "HELLO"
```

Add:

```
60 HOME
```

Note the effects of each command

*Use VTAB, HTAB, TAB, and PRINT commands in a loop, to create patterns on the screen. For example,

```
10 PRINT "***____***"
20 HTAB X
30 X=X+1
40 IF X=30 THEN GOTO 60
50 GOTO 10
60 END
```

This is a popular way to create interesting images. Students may wish to explore it at length.

*Write a program that will have only your name at the center of a clear screen. Now, make the name blink! (See sample programs BLINKING and BLINKING2. The trick is to use FOR-NEXT statements to time the length of the blinks and to alternate between printing the name and printing blanks.)

BEST COPY AVAILABLE

```
]LOAD BLINKING2
]LIST

10  REM :PROGRAM NAME:BLINKING2
15  TEXT
20  INPUT "WHAT WORD WOULD YOU LIKE TO BLINK? ";B$
25  PRINT
30  INPUT "ENTER A NUMBER BETWEEN 100 AND 1000.  THE LARGER THE NUMBER THE
      SLOWER THE BLINK. ";T
40  HOME
50 A$ = B$: GOSUB 100
60 A$ = "                          ": GOSUB 100
70  GOTO 50
100  VTAB 12: HTAB 18
110  PRINT A$
120  FOR I = 1 TO T: NEXT
130  RETURN

]
```

```
 PR#1
]LOAD BLINKING
]LIST

5  TEXT
10  VTAB 20
20  PRINT "MARTIN"
30  FOR I = 1 TO 120: NEXT I
40  VTAB 20
50  PRINT "      "
60  FOR I = 1 TO 120: NEXT I
70  GOTO 10

]
```

topic:FOR-NEXT Loops

filename:FORNEXT

RULES:

*The FOR-NEXT loop is the most efficient way for a program to count off a series of numbers and perform an operation on each one.

*The FOR-NEXT loop begins with the FOR statement and ends with the NEXT statement.

*The FOR statement tells the computer the range of numbers to use. The NEXT statement tells the computer to go back to the FOR statement and repeat the loop. For example, the following program causes the computer to print the numbers from 0 to 100.

```
10 FOR X = 0 TO 100
20 PRINT X
30 NEXT X
40 END
```

*When the program exhausts the range of numbers in the FOR statement, control passes to the line after the NEXT statement.

*The STEP command controls the steps the computer counts off for the loop. For example, the following program prints only the even numbers between 0 and 100.

```
10 FOR X = 0 TO 100 STEP 2
20 PRINT X
30 NEXT X
40 END
```

EXCERCISES:

*Modify the looping program above to count off by different units, for different ranges. Make it count backwards (with a negative STEP)!

*The following program makes the computer count from 1 to 1000 before printing "GOODBYE". Notice the delay between "HELLO" and "GOODBYE".

```
10 PRINT "HELLO"
20 FOR I = 1 TO 1000
30 NEXT I
40 PRINT "GOODBYE"
```

Modify the program so it waits a briefer period between printing words. Now, try a longer interval. Now, create as new program which uses a pause.

BEST COPY AVAILABLE

*Create a program which prints your name on the screen exactly 10 times. For example,

```
10 FOR I = 1 TO 10
20 PRINT "your name"
30 NEXT I
```

*Add the following line to the program in the previous exercise.

```
40 PRINT I
```

What do you think will be the result of the new line?

*Write a program to print the numbers from 1 to 1000. Use it to estimate how long it would take for the computer to print the numbers from 1 to 1,000,000. How about 1 to 1,000,000,000?

*Write a program to print out some arithmetic tables. For example,

```
10 PRINT "NUMBER","SQUARE"
20 FOR I=1 TO 20
30 PRINT I, I*I
40 NEXT I
```

topic: STRING Manipuation

file name:STRING

prerequisite: VARIABLES, PRINT

RULES:

*LEN(A$) returns the number of characters in the string A$. That number can be put into a variable, e.g., X = LEN(A$). If A$ contains a word that is 5 letters long, X = 5.

*LEFT$(A$,3) returns the 3 leftmost characters of a string. These must be used in a string variable like X$, not a numeric variable. Any number can be used instead of the 3, and any string variable name is acceptable.

*RIGHT$(A$,2) returns the 2 rightmost characters.

*MID$(A$,5) returns a "substring" starting at the 5th position of A$ and ending with the last character.

*MID$(A$,3,2) returns two charcters starting at position 3.

*The numbers specified in commands such as RIGHT, MID, and LEFT may be variables, e.g., RIGHT (A$, X).

*Strings may be CONCATENATED (joined together) thru the use of the plus (+) sign.

EXCERCISES: *Tell the computer that...

A$="STAR TREK"

Now tell the computer to print, as follows.

?LEN(A$)

?LEFT$(A$,3)

?RIGHT$(A$,1)

?MID$(B$,4,3)

Repeat the process with a different string, like...

B$="LUKE SKYWALKER"

Try the following.

C$="THE DUKES OF HAZZARD"

?LEFT$(B$,5)+LEFT$(C$,4)+MID$(C$,5,4)

Using parts of the strings already given, have the computer print STAR WARS.

*prequesite: FOR-NEXT, INPUT.
Examine the program THE. Modify it to search for a different word. Note that if the target word does not have three letters in it, the string manipulations will have to be changed.

*Run the program LIGHTS. Modify it to take a name from an input statement. Note that this requires the string manipulations to be modified, as well as adding an INPUT instruction. Modify the program REVERSE in a similar fashion.

BEST COPY AVAILABLE

```
]LOAD REVERSE
]LIST

5  REM   PROGRAM NAME:REVERSAL
10 A$ = "REVERSE"
15  PRINT A$
20 B$ =  RIGHT$ (A$,1) +  LEFT$ (A$,6)
35 A$ = B$
40  FOR X = 1 TO 100: NEXT X
50  GOTO 15

]
```

```
]LOAD LIGHTS
]LIST

5  REM  PROGRAM NAME IS LIGHTS
6  HOME
10 A$ = "YOUR NAME IN LIGHTS "
20 B$ =  RIGHT$ (A$,1) +  LEFT$ (A$,39)
25  VTAB 12
30  PRINT B$
32  REM  LINE 35 IS FOR THE DELAY
35  FOR X = 1 TO 90: NEXT X
40 A$ = B$
50  GOTO 20
```

```
 LOAD THE
]LIST

5  REM THE
10  REM :FROM COMPUTE JULY,81 PAGE 70
20  PRINT : PRINT "TYPE IN A SENTENCE. "
25  PRINT : PRINT
26  INPUT A$
27  PRINT
30  FOR W = 1 TO  LEN (A$) - 2
40  IF  MID$ (A$,W,5) = " THE " THEN  PRINT "YOU USED THE WORD 'THE'": END

50  NEXT W
60  PRINT "YOU DID NOT USE THE WORD 'THE'."
70  GOTO 20

]
```

topic:RANDOM NUMBERS

filename: RANDOM NUMBERS

DEFINITION:A sequence of numbers is random if the next number in the sequence cannot be predicted.

RULES:
*The instruction RND(1) returns a number greater than 0 and less than 1, i.e., a decimal fraction between 0 and 1. Contrary to expectations, RND(2) has exactly the same effect. It produces random numbers between 0 and 1. The same holds for RANDOM(3), RANDOM(4), etc. Thus, there is no reason to use any number other than 1.

*The INT command is useful in conjunction with RANDOM. It returns the numbers to the left of the decimal point, e.g., INT (3.526) returns 3. Therefore, INT(10*RND(1)) produces numbers between 0 and 9.

EXCERCISES:
*Run the following program.

```
10 PRINT INT(12*RND(1)+1)
20 GOTO 10
```

To see how this command works as it does, try the following:

```
10 PRINT RND(1)
20 GOTO 10
```

As you can see, this generates random numbers from 0 to 1, given to several decimal places. Now try this:

```
10 PRINT 12*RND(1)
20 GOTO 10
```

The above program gives random numbers from 0 to 11.99999999. Now, try the following program. It truncates the decimals, giving random numbers from 0 to 11:

```
10 PRINT INT(12*RND(1))
20 GOTO 10
```

To get numbers from 1 to 12 we must add the final 1.

```
10 PRINT INT(12*RND(1)+1)
20 GOTO 10
```

*The following program produces a table demonstrating the steps in manipulating random numbers. Do you understand it?

```
10 A=RND(1)
20 B=12*A
30 C=INT(B)
40 D=C+1
```

```
50 PRINT A,B,C;":";D
60 GOTO 10
```

*prerequisites:HOME, VTAB, HTAB. Write a program to place a character at a random position on the screen. Be particularly alert for the ILLEGAL QUANTITY ERROR message. What might cause it in a program like this? Now, modify the program so it repeats and see how long it takes to fill the screen. (See RNDGRID for sample solution.)

*prerequisite: IF-THEN. Did you notice in the previous exercise that whenever a character was printed in the lower right corner of the screen, the screen scrolled up one line? Can you modify your program to prevent this?

*prerequisite: ARRAYS
Run the program RND. This program generates 1000 random numbers from 1 to 10, counts the number of occurences of each number and prints the totals. On the average, each number should occur 100 times. Run the program several times to see if this occurs. Rewrite the program to count to a different number. How many times should each occur out of 1000? Does it work?

*Write a program to simulate 100 rolls of two dice. Find the sum of each roll. Use an array to count the number of times each sum (1 to 12) occurred. (See DICE for sample solution.) Modify the program to also print the number of times each number (1-6) occurred for each die.

```
]LOAD RNDGR
]LIST

3  REM :PROGRAM NAME:RNDGRID
5  HOME
10 X =  INT (40 *  RND (1) + 1)
20 Y =  INT (24 *  RND (1) + 1)
40  HTAB X: VTAB Y
50  PRINT 1;
60  GOTO 10
```

```
]LOAD RND
]LIST

10  REM  PROGRAM NAME RND
15  REM  THIS PROGRAM GENERATES 1000 RANDOM NUMBERS FROM 1 TO 10.  IT COUN
     TS EACH OCCURENCE OF EACH NUMBER AND PRINTS THETOTAL.
20  DIM A(10)
25  FOR N = 1 TO 1000
30 X =  INT (10 *  RND (1) + 1)
40  PRINT X;",";
45  IF N = 1000 THEN  PRINT
50 A(X) = A(X) + 1
60  NEXT N
70  FOR B = 1 TO 10
80  PRINT B;" OCCURRED ";A(B);" TIMES."
90  NEXT B

]
```

```
 LOAD DICE
]LIST

5  REM :PROGRAM NAME:DICE
10  DIM COUNT(12)
20  FOR X = 1 TO 100
30 D1 =  INT (6 *  RND (1) + 1)
40 D2 =  INT (6 *  RND (1) + 1)
50 T = D1 + D2
60 COUNT(T) = COUNT(T) + 1
70  NEXT X
80  FOR N = 2 TO 12
90  PRINT N;" OCCURRED ";COUNT(N);" TIMES
100  NEXT N

]
```

topic:READ-DATA

filename:READ-DATA

RULES:

*If a program contains a READ statement, it must elsewhere contain a DATA statement.

*The READ statement takes items of data from the DATA statement and puts them into variables. For example, READ A,B,C takes 3 pieces of data from the data statement and puts them into the variables A, B, and C.

*The DATA statement supplies data to the variables listed the READ statement. Items of data must be separated by a comma.

EXCERCISES:

*Run the program READ-DATA1. Do you understand how it works?

*Can you design a program that does not stop with the OUT OF DATA ERROR message? See READ-DATA2 for a sample solution.

*Design a program which reads more than one number at a time. See READ-DATA3 for an example.

*prerequisite: STRINGS. Design a program which READS and manipulates both STRING and NUMERIC data. See READ-DATA4 for a sample solution.

*prerequisite:INPUT. Run the MAD-LIBS2 program. Have students create their own version of MAD-LIBS.

```
 LOAD READ-DATA2
]LIST

5  REM    PROGRAM NAME:READ-DATA2
10  READ NUMBER
15  IF NU = 9999 THEN  END
20  PRINT NUMBER
30  GOTO 10
40  DATA   1,5,44,3.3,100,9999

]
```

```
]LOAD READ-DATA1
]LIST

5  REM  PROGRAM NAME:READ-DATA1
10  READ NUMBER
20  PRINT NUMBER
30  GOTO 10
40  DATA  1,5,44,3.3,100

]
```

```
 LOAD READ-DATA3
]LIST

5  REM  PROGRAM NAME READ-DATA3
10  READ N1,N2
15  IF N1 = 9999 THEN  END
20  PRINT N1,N2
30  GOTO 10
40  DATA  1,2,3,4,5,6,7,8,9999,0

]
```

```
 LOAD READ-DATA4
]LIST

5  REM  PROGRAM NAME READ-DATA4
6  REM  THIS PROGRAM PRINTS THE AVERAGE OF TEST SCORES
10  READ NAME$,S1,S2,S3,S4
20  IF NAME$ = "MORK" THEN  END
30  PRINT NAME$;"'S AVERAGE IS ";(S1 + S2 + S3 + S4) / 4
40  GOTO 10
50  DATA   MINDY,100,100,100,100
60  DATA   MR. BIXBY,45,67,65,81
70  DATA  MORK,0,0,0,0

]
```

```
 LOAD MAD-LIBS2
]LIST

3  REM  PROGRAM NAME MAD-LIBS2
5  PRINT "LETS PLAY MAD-LIBS!"
6  PRINT "I'LL ASK YOU FOR SOME WORDS AND THEN SUBSTITUTE THEM FOR WORDS I
     N A FAMOUS SPEECH."
10  PRINT "PLEASE TYPE IN A PLURAL NOUN"
20  INPUT N1$
30  PRINT "PLEASE TYPE IN ANOTHER PLURAL NOUN."
40  INPUT N2$
50  INPUT "NOW, AN ADJECTIVE: ";A1$
60  INPUT "A SINGULAR NOUN: ";N3$
70  INPUT "A PLURAL NOUN: ";N4$
80  INPUT "AND FINALLY, AN ADJECTIVE: ";A2$
85  PRINT : PRINT
90  READ S$: PRINT S$;N1$;
100  READ S$: PRINT S$;N2$;
110  READ S$: PRINT S$;A1$;
120  READ S$: PRINT S$;N3$;
130  READ S$: PRINT S$;N4$;
140  READ S$: PRINT S$;A2$
200  DATA  FOURSCORE AND SEVEN ," AGO, OUR "
210  DATA  " BROUGHT FORTH UPON THIS CONTINENT A ", " "
220  DATA  " DEDICATED TO THE PROPOSITION THAT ALL "," ARE CREATED "

]
```

topic: ARRAYS

filename: ARRAYS

PREREQUISITES: FOR-NEXT statements.

RULES:

*An array is a table of numbers or strings.

*DIM A(20) sets aside 21 elements (0 thru 20) for the the array "A". (Any number can replace the 20.) If an array element is used in a program before the DIM statement, APPLESOFT will assign a maximum subscript of 10.

*A(5) is the fifth element of a numeric array. A$(6) would be the sixth element of an array of strings. A variable can replace the number, e.g., A$(N) would be the eighth element of a string array, if N=8.

EXCERCISES:

*Study and RUN the program ARRAY1. Do you undersand how it works? Predict what will happen if you now enter the command...

```
PRINT A$(1)
```

*Write a more efficient version of ARRAY1 by replacing lines 80 to 160 with a FOR-NEXT loop. (See ARRAY2B for a solution.)

*Study and RUN the program ARRAY2. Predict what will happen if you then enter the following commands...

```
N=3
PRINT A$(N)
```

*Create a Mad-Libs or poetry program. Write it so that the user inputs nouns, verbs, etc. Have each type of input placed into an array. Then read from the arrays randomly, to produce the output.

```
 LOAD ARRAY1
]LIST

10  REM :PROGRAM NAME: ARRAY1
20  PRINT "THIS PROGRAM PRINTS THE ELEMENTS OF AN ARRAY."
25  PRINT
30 A$(1) = "ONE, TWO, BUCKLE MY SHOE"
40 A$(2) = "THREE, FOUR, SHUT THE DOOR"
50 A$(3) = "FIVE, SIX, PICK UP STICKS"
60 A$(4) = "SEVEN, EIGHT, LAY THEM STRAIGHT"
70 A$(5) = "NINE, TEN, A GOOD FAT HEN"
80  PRINT A$(1)
90  PRINT
100  PRINT A$(2)
110  PRINT
120  PRINT A$(3)
130  PRINT
140  PRINT A$(4)
150  PRINT
160  PRINT A$(5)

]
```

```
 LOAD ARRAY2B
]LIST

10  REM :PROGRAM NAME: ARRAY1
20  PRINT "THIS PROGRAM PRINTS THE ELEMENTS OF AN ARRAY."
25  PRINT
30 A$(1) = "ONE, TWO, BUCKLE MY SHOE"
40 A$(2) = "THREE, FOUR, SHUT THE DOOR"
50 A$(3) = "FIVE, SIX, PICK UP STICKS"
60 A$(4) = "SEVEN, EIGHT, LAY THEM STRAIGHT"
70 A$(5) = "NINE, TEN, A GOOD FAT HEN"
80  FOR I = 1 TO 4
90  PRINT A$(I)
100  PRINT
110  NEXT I

]
```

```
 LOAD ARRAY2
]LIST

5  REM :PROGRAM NAME: ARRAY2
10  PRINT "THIS PROGRAM SETS UP A ONE DIMENSIONAL ARRAY WITH FOUR ELEMENTS
     ."
20  PRINT
30  FOR N = 1 TO 4
40  PRINT "ENTER A WORD"
50  INPUT A$(N)
60  NEXT N
70  PRINT
80  FOR N = 1 TO 4
90  PRINT "A$(";N;")=";A$(N)
100  NEXT N

]
```

TOPIC: NESTED LOOPS

filename: NESTS

PREREQUISITE: FOR-NEXT statements

RULES:

*A NESTED LOOP is a loop within a loop.

*A NESTED LOOP generally is constructed with two FOR-NEXT statements. For example, look at the listing for the program NLOOPS1. Lines 10 and 50 form an outer loop. Lines 20 thru 40 form another loop. (The colons are there to improve readability and are ignored by the computer.) When this program is run, X is set to 0 and then the inner loop is executed for values of Y from 0 to 5. X then becomes 1 and the inner loop is again executed. This continues until the outer loop has been completed (i.e., X=2).

EXCERCISES:

*Run the program NLOOPS1. Try running the program with lines 40 and 50 interchanged. What is the result? Now, run the original program with line 10 interchanged with 20 and 40 interchanged with 50. What is the result?

*prerequisite: ARRAYS

Run the program NLOOPS2. This program makes use of arrays. Which statements caused the adjectives to be paired with the nouns and printed? (Hint: they form a NESTED LOOP.) Now, enter the following commands:

```
PRINT ADJECTIVE$(2)
PRINT AD$(1)
?NOUN(3)
?NO(4)
```

Why don't we have to type the whole name for each variable? The computer has tagged and memorized each word in the DATA statements. Which lines of the program did this?

*What is the series AD$(1), AD$(2)....called?

*Use the RND command to create a program which reads strings RANDOMLY from an ARRAY. This program can be expanded to become a poetry writing device.

*Use nested loops to create a program which PLOTS each point on the graphics screen in an orderly fashion. Change the color after each PLOT to create an intriguing pattern.

*To create an even more complex visual disply, create a program which plots more than one point each time it loops. for example, it could plot points on opposite sides of the screen during each loop.

```
 LOAD NLOOPS2
]LIST

10  REM  PROGRAM NAME NLOOPS2
20  FOR I = 1 TO 3
30  READ ADJECTIVE$(I)
40  NEXT I
50  FOR I = 1 TO 4
60  READ NOUN$(I)
70  NEXT I
80  FOR I = 1 TO 4
90  FOR J = 1 TO 3
100  PRINT AD$(J)" "NO$(I)
110  NEXT J
120  NEXT I
130  DATA  FUNNY,WEIRD,SILLY
140  DATA  BLIP,BOOP,BLOP,BLAP

]
```

```
 LOAD NLOOPS1
]LIST

5  REM  PROGRAM NAME:NLOOPS1
6  REM
7  REM  THE COLONS IN LINES 20 THRU 40 ARE NOT NECESSARY AND ARE IGNORED I
     N THE COMPUTER.  THEY ARE THERE TO ILLUSTRATE THE INNER LOOP WHICH IS
      NESTED WITHIN THE OUTER LOOP FORMED BY LINES 10 AND 50
8  REM
9  PRINT "X","Y": PRINT
10  FOR X = 0 TO 2
20 : FOR Y = 0 TO 5
30 : PRINT X,Y
40 : NEXT Y
50  NEXT X

]
```

topic: FLOWCHARTS

filename: FLOWCHARTS

RULES:

*It's good practice to plan a program with paper and pencil before going to the computer.

*A flowchart is one way to plan a computer program--or any series of steps to solve a problem.

*A flowchart is a drawing of the steps used to solve a problem. It shows the order of the steps and how they are connected to each other.

*A flowchart does NOT give the details of how each step is to be performed.

*An important part of a flowchart is its branches. A branch occurs when there is a choice between steps in the problem-solving sequence. Which way to turn depends on the answer to a question.

*The shapes that are used to indicate each step in a flowchart are arbitrary, but standard. (See the next pages for examples.)

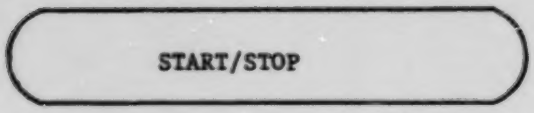
EXERCISES:

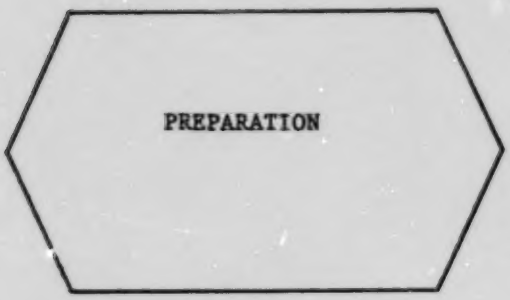
*Make flowcharts of everyday activities such as brushing teeth (see the next pages for example), tuning in a favorite t.v. show, washing the dinner dishes, etc.

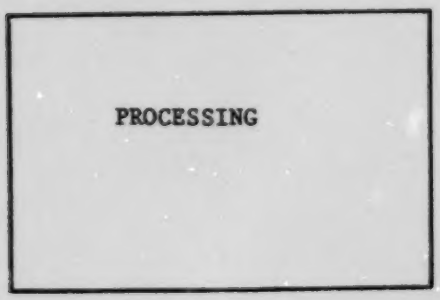
*Make flowcharts of one or more programs from this manual: DIAG1, DIAG2, LUCKY NAME, GUESS (see examples), RND, and/or DICE.

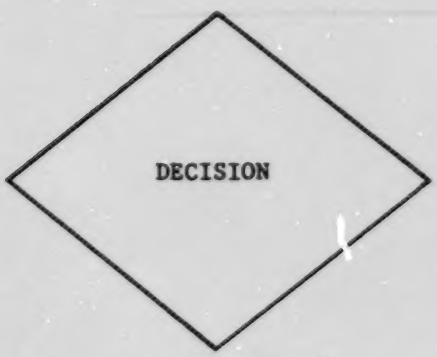
*Decide on an objective for a computer program. Then, write a flowchart showing its steps. Finally, write a program based on the flowchart. Revise the flowchart as you debug the program.

Standard Flowchart Symbols

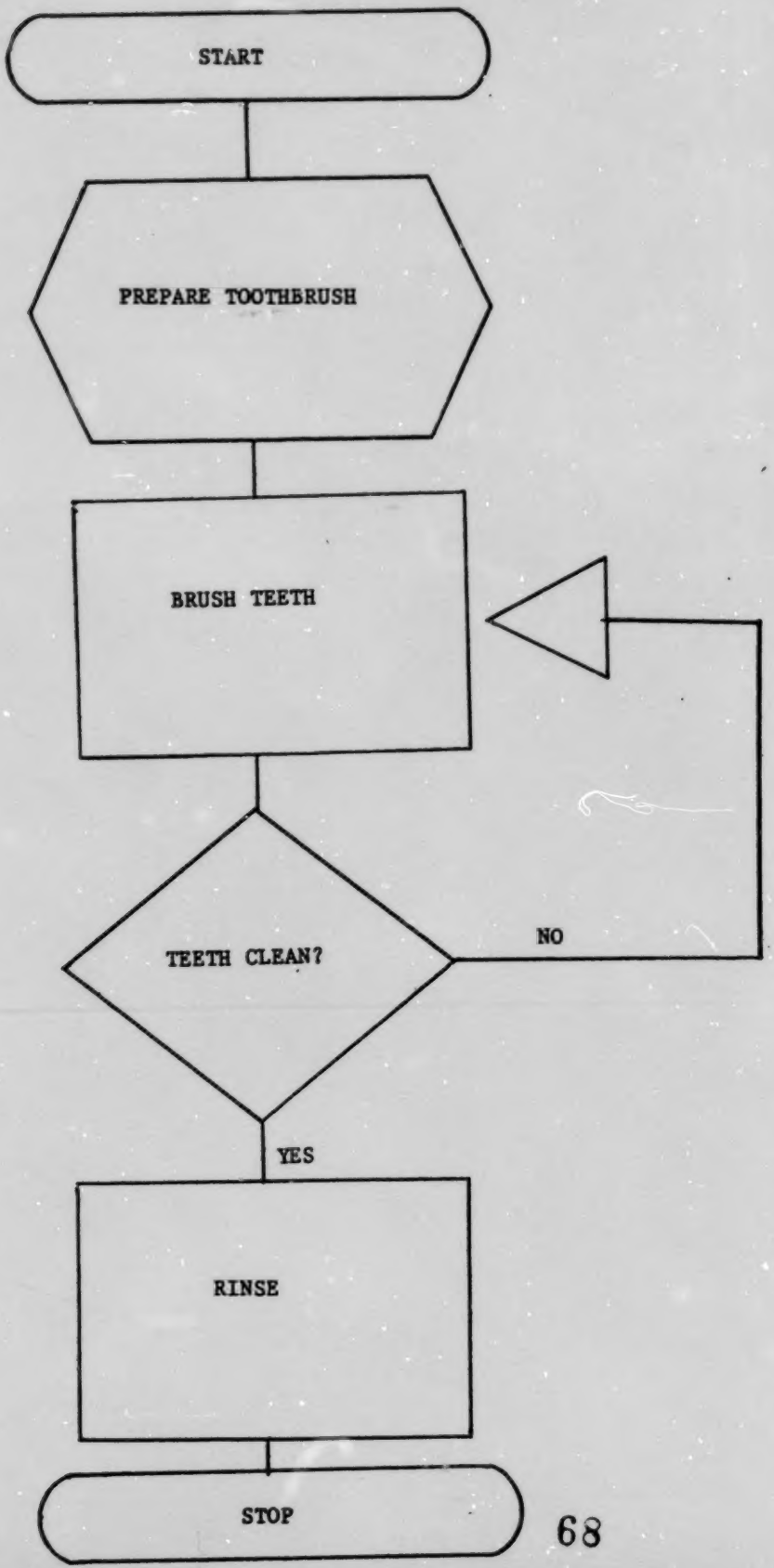




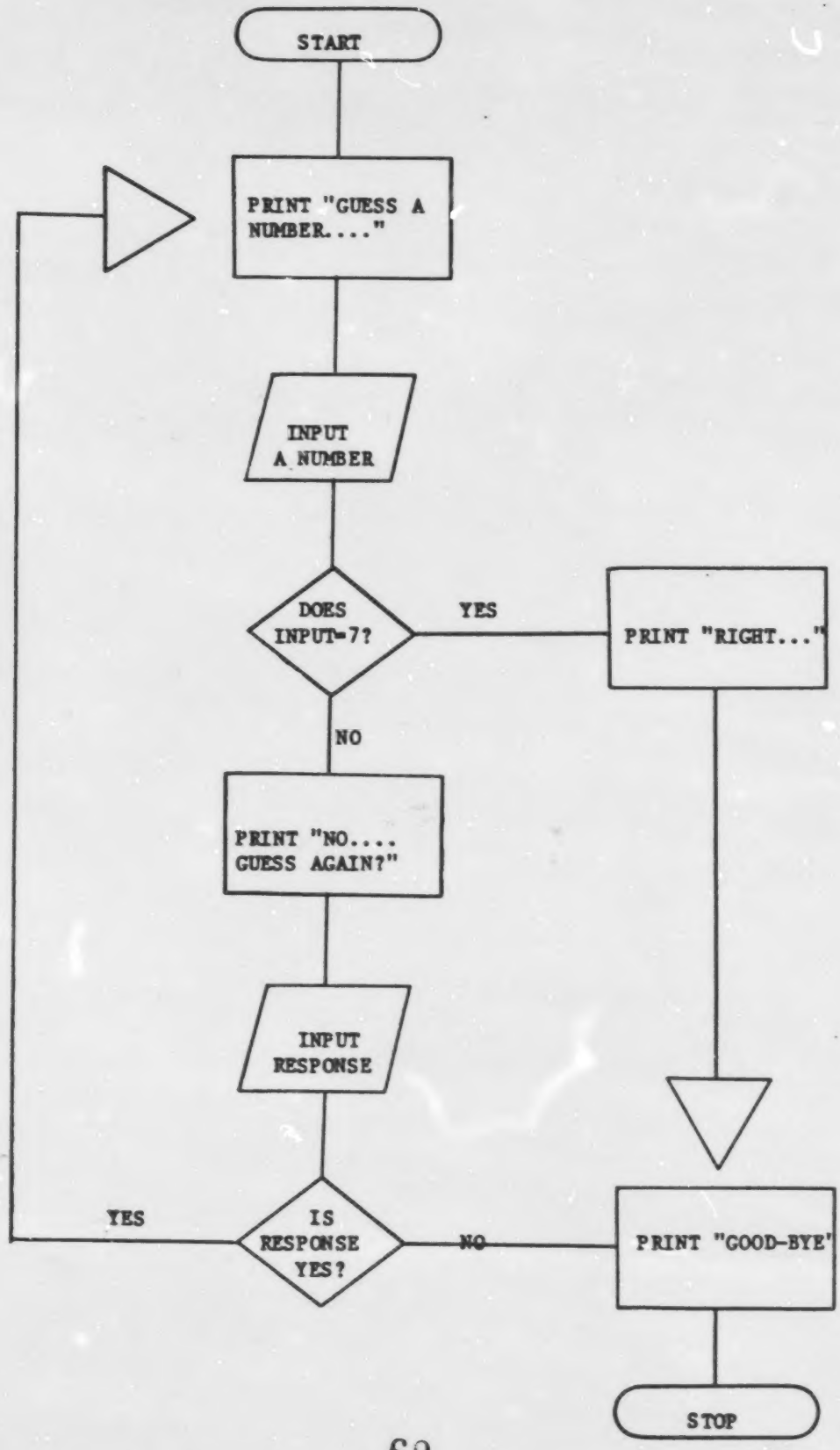




FLOWCHART FOR BRUSHING ONE'S TEETH



Flowchart of program GUESS



topic: COMPUTER ACTIVITIES WITHOUT THE COMPUTER

filename:NOCOMPUTER

RULES:

*Most computer operations are mechanical versions of human thought processes. So, they can be easily imitated by human beings.

EXERCISES:

*This activity lets children "walk through" any computer program. You will need the following materials:

5 empty boxes (cigar, shoe, etc.)
1 package of index cards
5 pens or pencils
5 labels for the boxes
1 marking pen
2 small blackboards with chalk and erasers

SET UP as follows: label the boxes with the letters A, B, C, D and E. Title one blackboard at the top as "The Screen (output)". The empty boxes represent the variables (which are empty until a value is stored in them). You will use one blackboard to write down a computer program and the other to write down the results (output) of the program.

Here is the ACTIVITY:

1. Begin by having the class help you write a BASIC program on the board. For example,

```
10 LET A = 10
20 LET B = 2
30 PRINT A;" TIMES ";B;" IS ";A*B
40 END
```

2. Divide the class as follows:
   —at least two students at each box. (One will be the variable and one will be the "bus." The bus is a device which transports information from one location to another in the computer.) Place a stack of index cards and pen or pencil near each box.
   —one student at "the screen" with chalk and eraser.
   —remainder of class at some central location, as "the central processing unit" (CPU), with a stack of index cards and pen or pencil.

3. Run the program:
   —the LET statement instructs a variable to write something (e.g., a number) on an index card and put it in a box. This is called the VALUE of the variable. If another value is already in the box, it is replaced.
   —CALCULATIONS are performed by the central processing unit.
   —The "busses" bring values from the variables to the CPU for calculations and return the results to their proper variables. The busses should make copies of the values, rather than taking the variable's copy.

—The busses also bring values to the blackboard to execute PRINT statements.

4. As the students learn new commands in BASIC, devise new procedures to represent them with variables, busses, the CPU, and the screen.

5. Here's the way the program listed above might run:
—students write the number 10 on an index card and place it in variable box labelled B
—students write the number 2 on an index card and place in variable box labelled B
—"Busses" bring the values of A and B (10 and 2, respectively) to the CPU. CPU writes the value of A*B (which is 20) and sends all three values to "screen", where a student writes the following: "10 TIMES 2 EQUALS 20".

*This type of activity can be implemented for Logo as well as BASIC. In Logo, you will probably utilize "turtle graphics". An area of floor covered with newsprint will make a more dramatic screen than a small blackboard. A pair of students can serve as the turtle, walking through, and drawing, the program's instructions, using values received

BEST COPY AVAILABLE

Bibliography

Albrecht, Robert. BASIC Self-Teaching Guide. Wiley, 1978.

Apple Computer Inc. Apple II Reference Manual, 1981.

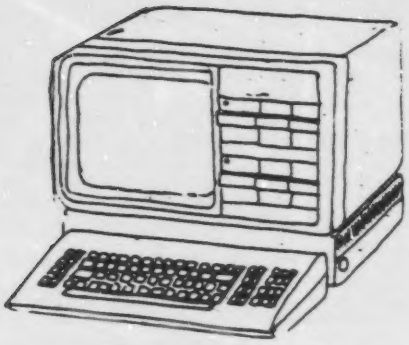
Apple Computer Inc. The Applesoft Tutorial. 1981.

Lipscomb, Susan Drake and Zuanich, Margaret Ann. BASIC Fun: Computer Games, Puzzles, and Problems Children Can Write. Avon, 1982.

Wilson, Mary Alice. Helping Teachers Become Computer Literate: A BASIC Approach. Hamshire Educational Collaborative, 1983.

BASIC Programming Activities for Secondary Students

by Patricia Taylor



Project Logic

Hampshire Educational Collaborative
South Hadley, MA 01075

TABLE OF CONTENTS — ACTIVITY NUMBER

Beginners: PRINT Statements 1

GOTO, CTRL C, Comma & Semicolon 2

INPUT Command 3

Variables 4

FOR-NEXT 5

ON__GOTO__ 6

Concepts of Probability 7

Writing a Program in the Form of a Multiple Choice Quiz (using several commands) 8

READ-DATA 9

Intermediate: READ-DATA 10

Arrays 11

CHR$ 12

Programming to Find Roots of a Polynomial. 13

LEFT, RIGHT, MID 14

Word Scramble Program (concatenation) 15

Jr/Sr High: Text Files, MON 16

Text Files 17

Reaching Objectives by Using Files 18

Writing a Library Program Using Text Files 19

Text Windows 20

Writing Programs Using a Counter 21

Algebra II 22

Graphics 23

Programs to Explore the World of Low Resolution Graphics 24

NECESSARY MATERIALS

All programs were written on an Apple Computer.

INSTRUCTIONAL OBJECTIVE

To write a BASIC program using the PRINT statement.

PROCEDURES

1. Review the print statement in the immediate execution mode and the use of ; in the PRINT statement.

2. Review the necessity of quotation marks for printing words instead of numbers.

3. Review the use of the words LIST and RUN.

4. RUN a demonstration program - this one is usually a hit.

```
10 HOME: VTAB6 : HTAB5 : SPEED = 150
20 ? "I'M HUNGRY!"
30 ? "PLEASE FEED ME - GIVE ME A PROGRAM"
40 SPEED = 255 : END
```

Some students may be very interested in lines 10 and 40. They are optional of course, but students interested in special effects often use these ideas immediately.

5. Have the class experiment with various PRINT statements.

Example I

```
10 ? "THIS IS A ONE LINE PROGRAM"
RUN
```

Example II

```
10 ? "MY NAME IS SUSAN"
20 ? "I LIVE IN NEW YORK"
30 ? 432
RUN
```

Example III

```
10 ? "THIS PROGRAM ADDS AND THEN MULTIPLIES TWO NUMBERS"
20 ? "THE NUMBERS ARE 2416 AND 317"
30 ? "THERE SUM IS"
40 ? 2416 + 317          (no quotation marks)
50 ? "THERE PRODUCT IS"
60 ? 2416 * 317          (no quotation marks)
```

Example III can be used later when students learn the INPUT command. It should also be modified now to show that the numbers in line 20 do not need to be enclosed in quotation marks.

```
20 ? "THE NUMBERS ARE "; 2416; "AND"; 317
```

Or if variables have been introduced -

```
15 A = 2416
17 B = 317
20 ? "THE NUMBERS ARE ";A; "AND" ; B
40 ? A + B
60 ? A * B
```

Tell students that line numbers do not have to be typed in order. The computer will automatically execute directions beginning with the lowest line number, sequentially continuing until the highest number is reached. Moreover, although many programs increment line numbers by ten, this is not mandatory. Be sure, however, to leave some room between lines, so lines which were forgotten or lines to modify and improve the program can be added later.

6. Here is a more involved program to introduce computers. You may RUN this one to get interest started.

```
 5 HOME
 6 VTAB6 : HTAB15
10 ? " HELLO THERE"
15 ?
16 HTAB8
20 ? "I AM ISAAC, YOUR COMPUTER"
25 ?
30 FOR I = 1 TO 4000 : NEXT I
40 HOME : VTAB6 : HTAB5
50 ? "DID YOU KNOW THAT I AM HUNGRY?"
60 ?
70 FOR W = 1 TO 3000 : NEXT W : ?
80 FOR I = 1 TO 50
90 Q = PEEK (-16336) + PEEK (-16336)      <-- sound
100 NEXT I
110 SPEED = 100 : HTAB5
120 ? "PLEASE FIGURE OUT A WAY TO FEED ME!"
130 FOR I = 1 TO 5000 : NEXT I
140 SPEED = 255
150 RUN
```

SUPPLEMENTARY ACTIVITIES

Have students write a letter using the PRINT command, or perhaps a mad lib.

Try to improve the program by changing the length of PRINT statements so words are not cut off at the end of the screen.

Write a program to show several math functions involving two numbers such as their sum, difference, product and quotient.

NOTES

Students may wonder if the computer "knows" or "remembers" their name, address, etc. Have them change names and run their programs. Shut the machine off and see what happens.

ACTIVITY 2
Beginners
GOTO Command
CTRL C
Comma & Semicolon

INSTRUCTIONAL OBJECTIVE

To write a simple program using the GOTO command.

To stop the RUN of a continuous loop.

To examine differences created by using , and ; within a program.

PROCEDURES

1. Write a two line program such as this.

```
10 ? "HA HA! YOU CAN'T STOP ME"
20 GOTO 10

RUN
```

or

```
10 ? "THIS IS WHERE I STARTED"
20 ? "SEND ME HERE"
30 GOTO 20
```

2. Instruct students on the use of CTRL C to stop a run.

3. Write a similar program using ;

```
10 ? "HERE I AM NOW ";
20 GOTO 10
```

A question of spacing usually results. Students often insert a space after the quotation mark, before ; -- Experiment with spacing before the H in HERE, after the W in now as well.

4. RUN the same program with a , after the quotation mark. RUN a program such as

```
10 ? "SPACING",
20 GOTO 10
```

On the apple, three columns are set up when a comma (not within quotes) is used, unless three won't fit due to the length of the statement.

5. Write several programs using numbers instead of words. See if the results are the same. Depending on the class , try the following.

```
10 ? "FOUR TIMES SIX IS "; 4*6
20 GOTO 10
```

or

```
10 ? "TEN X TEN IS ", 10*10
20 GOTO 10
```

Have students think of a difficult computation relative to what they are presently learning, PRINT it in line 10 and use the GOTO command in line 20.

Students like to see their name in print.

Try:

```
10 ?"SUE SMITH" ;
20 GOTO 10
```

then

```
10 ? "SUE SMITH" ,
20 GOTO 10
```

INSTRUCTIONAL OBJECTIVES

To incorporate the INPUT command in BASIC programs.

PROCEDURES

The INPUT command permits us to provide the computer with names (words) or numbers to work with. Whenever the computer encounters an INPUT command, it prompts us with a question mark while it patiently waits for us to reply.

1.
```
10 ? "I WILL ASK YOU FOR THREE NUMBERS, ONE AT A TIME."
20 ? "THE FIRST NUMBER IS ";
30 INPUT A
40 ? "THE SECOND NUMBER IS ";
50 INPUT B
60 ? "THE THIRD NUMBER IS ";
70 INPUT C
80 ? "THE SUM OF ";A;" AND ";B;" AND ";C;" is ";A+B+C
90 ? "THEIR PRODUCT IS ";A*B*C
```

2. Next, begin with simple questions -

```
10 ? "WHAT IS YOUR NAME?"
20 INPUT N$
30 ? "HELLO ";N$;" HOW ARE YOU TODAY?"
```

Point out that line 20 needs a $ because the response will be a word. It is also convenient to make a connection between the string and the response as N$ is for name, A$ for address, etc.

Line 30 may cause some problem for students who are confused and think that the computer knows the answer to each question. We can lie in line 20 and the computer "believes" us. It also asks another question, but without an INPUT for it, the program ends.

Explain that a $ symbol is needed for words, but is not necessary for numbers.

```
10 ? "HOW OLD ARE YOU?"
20 INPUT A
30 ? "REALLY, YOU DON'T LOOK ";A
```

If A$ were used the resulting program wouldn't look noticeably different but, without adjustment, the age obtained couldn't later be used in a computation.

3. Write several programs using the INPUT command. Shut the machine off after a run to show that the computer doesn't remember an answer to an INPUT question. Use the string obtained from an INPUT to show that until another RUN is made or the computer is shut off, the answer to INPUTs are the same and can be used again in the same program.

```
10 ? "HELLO THERE. WHAT IS YOUR NAME?"
20 INPUT N$
30 ? "THAT'S A BEAUTIFUL NAME.  IN FACT I WISH ";N$;
   " WERE MY NAME."
40 ? "HOW OLD ARE YOU?"
50 INPUT A
60 ? "WELL ";N$;" IT IS HARD TO BELIEVE YOU ARE ";A;"
   YEARS OLD."
70 ? "YOU DON"T LOOK A DAY OVER ";A-10
```

Line 70 will print negative answers if INPUTs are small (under 10). If 50 used A$, line 70 wouldn't work. The VAL command would be needed.

4. Show that an INPUT will not only accept and label a reply, but will PRINT too.

```
10 INPUT "WHAT YEAR IS IT? ";Y$
20 ? Y$ ;" ALREADY? WOW! TIME FLIES."
```

Line 10 can be used instead of -- 5 ? "WHAT YEAR IS IT?"
10 INPUT Y$

SUPPLEMENTARY ACTIVITIES

Have students write a letter (to Santa if Christmas is near), conduct an interview or write a mad lib or a poem using INPUT statements. Students generally love this assignment and work hard to make their program the best (often the funniest). Allow several periods for completion.

The next program also uses an IF -- THEN statement, which has not yet been formally introduced. Students quickly understand lines 70 and 80 even if they couldn't write these lines themselves at this point. If your computer has a name, this program is fun!

```
10 HOME : VTAB4 : HTAB6
20 ? "WHAT IS MY NAME?"
30 ? "              A. HARVEY"
40 ? "              B. ISAAC"
50 ? "              C. CLYDE"
60 INPUT R$
70 IF R$ = "B" THEN ? "YOU ARE REALLY ON THE BALL"
80 IF R$ <> "B" THEN? "HOW COULD YOU HAVE FORGOTTEN
   ME ALREADY?"
```

(explain this matching check →)

(<> is not equal)

NOTES

In the above program, no line is included for responding to INPUTs other than A, B, or C. Students should be aware of this. Most will be content with knowing their error of omission, others may want to handle the problem now.

```
65 IF R$ <> "A" OR R$ <> "B" OR R$ <> "C" THEN? "FOLLOW DIRECTIONS":
   GOTO 20
   (or GOTO 10)
```

ADDITIONAL PROGRAMS

1. A Letter to Santa

```
 10 HOME : VTAB6 : HTAB7
 20 ? "SANTA"S LETTER"
 30 ? : ? :?
 35 ? "PLEASE ANSWER THE FOLLOWING QUESTIONS"
 40 INPUT "WHAT IS YOUR NAME ";A$
 50 INPUT "TYPE AN UNUSUAL NAME ";B$
 60 INPUT "ENTER A TYPE OF ANIMAL ";C$
 70 INPUT "ENTER A NUMBER ";D$ (or D)
 80 INPUT "ENTER YOUR BEST FRIEND'S NAME ";E$
 90 INPUT "ENTER ANOTHER FRIEND'S NAME ";F$
100 INPUT "ENTER AN ADJECTIVE ";G$
110 INPUT "ENTER A NUMBER ";H$ (or H)
200 HOME
210 VTAB6 : HTAB4 : ? "DEAR SANTA,"
215 ? : ?
220 ?"     HI, MY NAME IS " ;A$;
230 ?"IS YOUR NAME REALLY ";B$;" ?";
240 ?"I HAVE A " ;C$;" NAMED ";B$; ".  ";
250 ?"BOY, ONLY " ;D$; " DAYS UNTIL CHRISTMAS,  AND I'M REALLY EXCITED. ";
260 ? E$;" AND ";F$;" ARE REALLY EXCITED TOO.";
270 ? "I ONLY WANT " ;H$; " GIFTS FOR CHRISTMAS THIS YEAR.";
280 ? "THAT"S PRETTY GOOD FOR A " ;G$; " KID LIKE ME."
290 ? "WELL, IT"S BEEN NICE TALKING TO YOU.  SEE YOU CHRISTMAS EVE!"
295 ? : ?
300 HTAB20 : ?"LOVE," : ?
310 HTAB20 ? A$
```

2. Mod Lib

```
10 HOME : VTAB(6) : HTAB(5)

20 INPUT "WHAT IS YOUR NAME?"; N$

30 INPUT  "WHO  IS YOUR FAVORTIE CELEBRITY?"; C$

40 INPUT "WHAT IS YOUR FAVORITE RESTAURANT?"; R$

50 INPUT "WHAT IS YOUR FAVORITE FOOD?"; F$

60 INPUT "WHAT IS YOUR FAVORITE COLOR?"; L$

70 INPUT "WHAT IS YOUR FAVORITE TYPE OF CAR?"; CR$

80 INPUT "NAME A GREAT PLACE"; PL$

90 HOME : VTAB(6) : HTAB(4)

100 ? "RONA BARRETT'S HOLLYWOOD REPORT"

110 HTAB(14) : ?"!!!FLASH!!!"

120 ?N$; "THE WELL KNOWN EX-CONVICT, WAS SEEN HAVING AN INTIMATE
    RENDEZ-VOUS WITH THE FAMOUS ";C$;".  THEY WERE HAVING DINNER ";

130 ?"FOR TWO IN THE ELEGANT RESTAURANT, ";R$;".  RUMOR HAS IT THAT THE
    TWO WERE SHARING A DISH OF ";F$;".  HOW ROMANTIC!"

140 ? "AFTER LEAVING THE RESTAURANT, ";N$;" AND ";C$;" DROVE AWAY IN"
    ;N$; "'S NEW, ";L$;" ";CR$;"."

150 ? "THEIR DESTINATION WAS UNKNOWN, BUT I HAVE HEARD RUMORS THAT THEY
    WERE OFF TO ";C$;"'S PENTHOUSE IN ";PL$

160 ?"TUNE IN NEXT TIME TO LEARN MORE ABOUT ";N$;" AND ";C$
```

3. A Program For Sequences -- Students must know algebra.

```
10 HOME : VTAB5 : HTAB4

20 ? "THIS PROGRAM WILL FIND THE N-TH TERM IN ANY ARITHMETIC PROGRESSION
     OR IT WILL IDENTIFY THE POSITION OF ANY TERM IN THE SEQUENCE.

30 INPUT "WHICH IS YOUR PREFERENCE? TYPE 1 TO FIND THE TERM, 2 TO FIND
   A POSITION IN THE SEQUENCE"; CS

40 IF VAL (CS) = 1 THEN GOTO 100

50 IF VAL (CS) = 2 THEN GOTO 200

60 GOTO 10

100 HOME : HTAB6 : VTAB7

110 ? "I WILL FIND ANY TERM IN AN ARITHMETIC PROGRESSION NOW"

120 INPUT "WHAT IS THE VALUE OF THE FIRST TERM"; F

130 INPUT "WHAT IS THE COMMON DIFFERENCE BETWEEN TERMS?";D

140 INPUT "WHICH TERM SHOULD I FIND FOR YOU";N

150 X = F +(N-1) * D

160 ?"THE ";N;"-TH TERM IN THE PROGRESSION IS "; X

170 INPUT "DO YOU WISH TO DO ANOTHER?"; R$

180 IF R$ = "YES" OR R$ = "Y" THEN GOTO 30

190 END

200 HOME : VTAB7 : HTAB6

210 ?"I WILL NOW FIND THE POSITION OF A CERTAIN TERM WITHIN AN ARITHMETIC
    SEQUENCE."

220 INPUT "WHAT IS THE FIRST TERM OF THE SEQUENCE?"; F

230 INPUT "WHAT IS THE DIFFERENCE BETWEEN TERMS?"; D

240 INPUT " NOW WHAT IS THE VALUE THAT I SHOULD IDENTIFY AS TO ITS POSITION
    IN THE SEQUENCE?"; X

250 N =(X-F)/D+1

260 IF N = INT(N) THEN ? "THAT VALUE CORRESPONDS TO THE";N; "-TH POSITION": GOTO 170

270 ? "SORRY, BUT YOU ARE NOT WORKING WITH AN ARITHMETIC SEQUENCE."
```

Have the class write a program for geometric sequences or for the sum of each type.

4. This Program Quizes Users On Vocabulary

```
10? "WHAT WORD MEANS POSSIBLE?"
20? "     A.  REASONABLE"
30? "     B.  PLAUSIBLE"
40? "     C.  ALTRUISTIC"
50 INPUT R$
60 IF R$ = "B" THEN ? "THAT'S RIGHT" : GOTO 100
70? "THE CORRECT ANSWER IS B. PLAUSIBLE"
100  -- next question --
```

5. Have students write a program that identifies the larger number of a pair. One method would be by using READ - DATA, another by using random generators.

```
10? "PLEASE TELL ME WHICH NUMBER IS LARGER. TYPE 1 IF THE FIRST IS
  LARGER, 2 IF THE SECOND IS LARGER."
20 X = INT (100 * RND(1)) + 1
30 Y = INT (100 * RND(1)) + 1
40? X; "                ";Y
50 INPUT R
60 IF X>Y AND R = 1 THEN ? "GREAT!"
70 IF Y>X AND R = 2 THEN ? "GREAT!"
80 ? "SORRY"
90 GOTO 20
```

Modify the program so it will END when the user INPUTs an "E. Then change the program so decimals are allowed, then change it to fractions. What midifications should be made for the possibility X = Y?

6. Guess A Number

```
10 ? "THIS IS A LITTLE GAME WHERE YOU HAVE TO GUESS THE NUMBER
   I'M THINKING OF."

20 ? "WHAT SHOULD THE LARGEST NUMBER BE?"

30 INPUT X

40 N = INT (X * RND(1)) + 1

50 ? "OK, I'M THINKING OF A NUMBER BETWEEN 1 AND ";X

60 ? "WHAT IS YOUR GUESS?"

70 INPUT G

80 IF G > N THEN ? "SORRY, BUT YOUR GUESS IS TOO LARGE." : GOTO 60

90 IF G < N THEN ? "NO, YOUR GUESS IS TOO LOW" : GOTO 60

100 ? "YOU GUESSED IT!  THE NUMBER WAS ";N : ?"DO YOU WISH TO PLAY AGAIN?" :
    INPUT R$ : IF R$ = "Y" OR R$ = "YES" THEN GOTO 20

110 END
```

Tell the class to change the program so that it reports how many tries were made before the correct guess was made.

Another change could be made in the beginning of the program; to enter both a lower and higher range for the computer.

A more difficult challenge would be to report on the magnitude of error made for each guess. If the difference is 50% away, one reply is made as opposed to another if the guess is only 5% off. Most classes initially try this adjustment by using numbers, as, if the differnece is 100 as opposed to a difference of 30. Once a small initial range for the computer is used, they quickly see the problem involved.

INSTRUCTIONAL OBJECTIVE

To understand variables in computer programming.

PROCEDURES

1. Explain that it can be inconvenient, if we are using constants often in a program, to write them each time. Or, if we wish to change the values throughout a program, it would be convenient to make adjustments to 1 line only. In this program, the computer identifies X as having a value of 10, Y as a value of 15. By changing 5 to X = 20; Y = 8 the computer will print 20, 8, 160.

```
 5 X = 10; Y = 15
10 ? X, Y, X * Y
```

2. Show the students the following program. See if they can predict the RUN. Although it would be easier here to print each number, the exercise does demonstrate the concept of variables.

```
 5 X = 1
10 ? X, X +1
20 ? "BUCKLE MY SHOE"
30 ? X + 2, X + 3
40 ? "KNOCK AT THE DOOR"
50 ? X + 4, X + 5
60 ? "PICK UP STICKS"
70 ? X + 6, X + 7
80 ? "LAY THEM STRAIGHT"
90 ? X + 8, X + 9
100 ? "A BIG FAT HEN"
```

3. Another program illustrating the point.

```
 5 HOME : VTAB5 : HTAB4
10 A$ = "WITH A NICK NACK PATTY WACK"
20 B$ = "GIVE THE DOG A BONE"
30 C$ = "[illegible]S OLD MAN CAME ROLLING HOME!"
40 D$ = "THIS OLD MAN, HE PLAYED"
50 E$ = "HE PLAYED NICK NACK ON MY"
60 INPUT "PLEASE ENTER A NUMBER LESS THAN 10"; X
70 INPUT "NOW ENTER A WORD THAT RHYMES WITH";X; R$
80 HOME : VTAB5
90 ? D$; X
100 ? E$; R$
110 ? A$ : ? B$ : ? C$
```

This program can be used later on when ready for a counter and substitution and endless loop:

add:

```
120 X = X + 1
130 GOTO 70          (This is an endless loop.)
```

Add line 125 IF X = [10] THEN END ([10] - or any other number)

To demonstrate a counter controlling the number of times a loop is executed, add line 75 Y=Y+1 : IF Y = 5 THEN END.

Change line 120 to X = X + 2 or X = X + 5 -- predict the results and RUN.

Ask the class to add a line to be sure the INPUT number is less than 10.

SUPPLEMENTARY ACTIVITIES

The following simple program can be used as a quiz, then RUN at the computer.

```
10 A$ = "HAPPY"
20 B$ = "SAD"
30 C$ = "MAD"
40 LET B$ = A$
50 LET C$ = B$
60 PRINT "I AM ";B$;".  YOU ARE ";C$
```

Here is a program that finds all factors of a number.

```
10 INPUT "PLEASE TYPE A NUMBER FROM 2 to 100"; X

20 IF X<2 OR X>100 THEN GOTO 10

30 REM*CHECK TO SEE IF X IS AN INTEGER

40 IF INT(X)<>X THEN ?"ENTER INTEGERS ONLY" : GOTO 10

50 FOR I = 1 TO SQR(X)

60 IF INT(X/I) = X/I THEN ? I " IS A FACTOR OF ";X :
   X/I  IS A FACTOR OF  ";X

70 NEXT I
```

Modify the program so the products will be shown in a chart:

Example:

```
YOUR NUMBER WAS 100

FACTORS ARE

    2 X 50
    4 X 25
    5 X 20
   10 X 10
```

INSTRUCTIONAL OBJECTIVE

Introduce the command FOR--NEXT and use in programs.

PROCEDURES

1. Explain to students, perhaps using a blackboard, the format of FOR--NEXT.

   I explain FOR--NEXT as a shoe box labeled with whatever letter is used after the FOR and containing just the numbers specified in the command.

   Example:

```
FOR X = 1 TO 30
NEXT X
```

   The numbers 1 through 30 would be arranged in order in a shoe box labeled X. When the computer reads the command NEXT, it goes to the box and chooses the next number..

   Do several easy programs such as this.

```
10 FOR X = 1 TO 10
20 ? X
30 NEXT X
```

   RUN this program. When finished ask the computer to PRINT X. X will have a value of 11. The fact the computer always increases the value of X by one after the FOR--NEXT is complete should be mentioned and made note of. If a program later uses X, students will wonder why 11, not 10, occured.

   Do several other programs like this one.

```
10 FOR A = 2 TO 100
20 ? 2 * A
30 NEXT A
```

   Introduce step functions: try the above with

```
10 FOR A = 2 TO 100 STEP 2
```

   or

```
10 FOR A = 100 TO 2 STEP-2
```

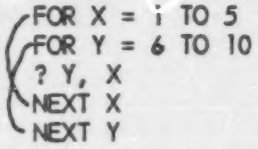
   Try decimal steps, too.

   Finally, try problems with more than one FOR--NEXT loop stressing that loops can not cross.

```
10 FOR X = 1 TO 5
20 FOR Y = 6 TO 10
30 ? Y, X
40 NEXT y
50 NEXT X
```

(This is an example of NESTING.)

Draw connectors between FOR--NEXT commands. If lines cross, the set up is wrong.



SUPPLEMENTARY ACTIVITIES

Have students PRINT a multiplication table using nested FOR--NEXT commands.

ADDITIONAL PROGRAMS

1. FOR--NEXT Commands

```
 5 SPEED = 150
10 ? "THIS PROGRAM WILL DEMONSTRATE THE FOR--NEXT COMMAND."
20 ? "FOR X = 1 TO 50, PRINT X, NEXT X GIVES US THE FOLLOWING"
30 FOR X = 1 TO 50
40 ? X
50 NEXT x
60 ? "NOW I'LL CHANGE, AND TRY FOR X = 1 TO 50 STEP 5, PRINT X,
   NEXT X.  SEE?"
70 FOR X = 1 TO 50 STEP 5
80 ? X
90 NEXT X
100 ? "I CAN COUNT IN REVERSE AS WELL.  WATCH ME PERFORM.  FOR X =
    50 TO 1 STEP -2."
110 FOR X = 50 TO 1 STEP -2
120 ? X
130 NEXT X
140 ? "I EVEN KNOW DECIMALS. CAN YOU DO THIS AS QUICKLY?  FOR X =
    100 TO 120 STEP .7, ? X, NEXT X."
150 FOR X = 100 TO 120 STEP .7
```

(continued..)

```
160  ? X

170 NEXT X

180 ? "ARE YOU IMPRESSED BY MY CAPABILITIES?  I THINK YOU'RE OK
    TOO."
```

2. Review prime numbers and try to write a program which will identify a number as prime or not.

```
10 HOME

20 INPUT " PLEASE INPUT A NUMBER AND I'LL TELL YOU IF IT IS PRIME"; X

30 FOR A = 2 TO X - 1

40 IF INT (X/A) = X/A THEN ? "THAT IS NOT A PRIME NUMBER.  IT CAN
   BE DIVIDED BY ";A;" BY";X/A; " AND PERHAPS MORE NUMBERS." : END

50 NEXT A

60 ? "THAT IS A PRIME NUMBER.  WANT TO TRY ANOTHER?"

70 INPUT R$

80 IF R$ = "Y" OR R$ = "YES" THEN GOTO 10

90 END
```

Change the program so it identifies the greatest factor of any number that is not prime.

```
30 FOR A = X - 1 TO 2 STEP -1

40 IF INT(X/A) = X/A THEN GOTO 100

100 ? A; " IS THE GREATEST FACTOR OF "; X
```

Modify the program so all factors of X are printed out in pairs. Now eliminate all duplicates.

3. This program quizes the user on simple addition problems and then reports the score.

```
10 S = 0
20 INPUT "HI THERE!  WHO ARE YOU?";A$
30 ? "WELL ";A$;" LET'S DO SOME ADDITION PROBLEMS."
40 FOR N = 1 TO 10
50 ? N;")";
60 X = INT (10 * RND (1)) + 1
70 Y = INT (10 * RND (1)) + 1
80 C = X + Y
90 ? X;" + ";Y;" = ";
100 INPUT B
110 IF B = C THEN ? "THAT'S RIGHT ";A$ : S = S + 1 : GOTO 160
120 ? TRY AGAIN."
130 INPUT Q
140 IF Q = C THEN ?"NOW YOU'RE COOKING." : S = S + 1 : GOTO 160
150 ? "THE CORRECT ANSWER IS "; C
160 NEXT N
170 S = 10 * S
180 ? A$; " YOUR SCORE IS ";S
```

The user is allowed two tries to get the correct answer and scores a point whether he is successful on the first attempt or the second. Change the problem so points are awarded only if the first response is correct. Now, change from addition to multiplication questions.

4. Grade Average

```
10 S = ø
20 INPUT "HOW MANY GRADES DO YOU WISH TO AVERAGE? ";N
30 DIM G (N)
40 FOR A = 1 TO N
50 INPUT "GRADE ";A; " IS "; G(A)
60 S = S + G(A)
70 NEXT A
80 ? "YOUR AVERAGE IS ";S/N
```

Re-write this program using READ--DATA.

*Try to write a program based on the final grade average you desire for a course. Compare the grades you have already received in the course and find the average you must get on the remaining tests to end up with your desired grade. Teacher should provide information as to how many tests are left.

INSTRUCTIONAL OBJECTIVE

To write a program using a random generator and the ON ___ GOTO ___ command.

PROCEDURES

1. Review the random generator process:

   - IF X = RND (1) then numbers from .0000 to .9999 are generated.
   - Change to: X = 6 * RND (1) the numbers are .0000 to 5.9999.
   - By adding the integer function to the command so that it is now X = INT (6 * RND (1)) we get numbers between 0 and 5 (whole numbers).
   - X = INT (6 * RND (1)) + 1 would result in an integer between 1 and 6, as in the tossing of a die.*

   Here is a brief program to simulate the tossing of a die 50 times.

```
 5 C = Ø
10 X = INT (6 * RND (1)) + 1 : REM**INTEGERS 1 - 6
20 C = C + 1 : REM** COUNTER
30 ?"THE LUCKY NUMBER IS ";X
40 IF C< 50 THEN GOTO 10
50 END
```

*If students are confused, have them run two line programs such as the following changing various parts of line 10.

```
10 Y = INT (98 * RND (1)) + 1
20 ?Y : GOTO 10
```

or

```
 5 A = 32
10 Y = INT (A * RND (1)) + 1
20 ? Y : GOTO 10
```

2. Next write a simple program using the ON ___ GOTO ___

```
 10 Y = INT (5 * RND (1)) + 1
 20 ? Y
 30 ON Y GOTO 100, 200, 300, 400, 500
100 ? "ONE, ONE, ONE"
200 ? "HERE IS A TWO"
300 ? "3!!!"
400 ? "FOUR NOW OR FOR NOW?"
500 ? "FIVE HIVES"
```

RUN it now and note the problems. Add lines 110, 210, 310, 410, 510 -- Have each read END or GOTO 10 depending on whether you want one choice or an endless loop.

Point out to students that there is no relationship between the line number and the random choice. That is, 300 didn't have to be the line choice for a random 3.

Line 30 could be ON Y GOTO 40, 62, 87, 98, 50 (of course, the rest of the program would be changed also).

There is another important point, but it's discussion shall be postponed until after this next program. See NOTES. *

SUPPLEMENTARY ACTIVITIES

Have the class write a program to simulate the tossing of dice.

The question of possibly using one random generator instead of two generally arises. The difference between X = INT (11* RND (1)) + 2
and X = INT (6 * RND (1)) + 1
Y = INT (6 * RND (1)) + 1
and using X + Y, is that the first does generate integers between 2 and 12, as dice would, but, not in the same manner. The probabilities when using one generator all equal 1/12. When using two and finding their sum, the probabilities differ and mirror the true tossing of two die.

Here is a program for tossing dice 1000 times.

```
  5 C = Ø
 10 X = INT (6 * RND (1)) + 1
 20 Y = INT (6 * RND (1)) + 1
 30 C = C + 1
 35 IF C = 1001 GOTO 1300
 40 ? X;"    ";Y;"    ";X + Y
 50 ON X + Y GOTO 100, 200, 300, 400, 500, 600, 700,
    800, 900, 1000, 1100, 1200
100 GOTO 10
200 TWO = TWO + 1 : GOTO 10
300 THR = THR + 1 : GOTO 10
400 FOUR = FOUR + 1 : GOTO 10
500 FIVE = FIVE + 1 : GOTO 10
600 SIX = SIX + 1 : GOTO 10
700 SVN = SVN + 1 : GOTO 10
800 EGT = EGT + 1 : GOTO 10
900 NINE = NINE + 1 : GOTO 10
1000 TEN = TEN + 1 : GOTO 10
1100 ELN = ELN + 1 : GOTO 10
1200 TWL = TWL + 1 : GOTO 10
1300 ?"THE PROBABILITIES FOR EACH WERE AS FOLLOWS"
1310 ?"FOR 2   "; TWO/C
1320 ? "FOR3   "; THR/C
etc.↓
```

Using C as 1000 shows a large sample. Change C and discuss the results, especially when C is a small number.

In line 50 point out the needed dummy line 100. Change line 50 by deleting the 100 and use TRACE to follow the lines as they are processed. Where does a result of 2 for a sum send the program if the 100 is not there?

NOTES

*Line 100 if not included would change 50 to ON X + Y GOTO 200, 300, 400,..... then, on the result of a random sum of 2, the second listing of line 50 would be chosen, that is, GOTO 300 (which is for sums of 3.)

ACTVITY/WORKSHEET

One intelligent and ambitious student wrote a program which showed the actual sum of the dice by graphing horizontal lines. As the program continued the horizontal lines formed a normal distribution curve. The visual effect was very impressive and convincing. We had already discussed the probability of tossing each number 2 through 12 and seeing the results, graphically represented, reinforced the math concepts involved. The program then switched to text and showed the mean, median, mode and standard deviation.

ADDITIONAL PROGRAM

This program uses a random number generator and a FOR--NEXT loop to arrange a list in order.

```
  5 DIM N (20)
 10 FOR I = 1 TO 20
 20 N (I) = INT (100 * RND (1)) + 1 : ?N(I) : REM**SHOW THE NUMBERS
 30 NEXT I
 40 REM**WE NOW HAVE 20 NUMBERS, BETWEEN 1 AND 100, WHICH WERE
    RANDOMLY SELECTED
 50 T = 0
* 60 FOR S = 1 TO 19
 70 IF N(S)< = N(S + 1) THEN GOTO 200
 80 LET A = N(S)
 90 LET N(S) = N(S = 1)
100 LET N(S + 1) = A
110 REM**A SWITCH HAS BEEN MADE SINCE THE ORDER WAS INCORRECT
120 T = 1
200 NEXT S
210 IF T = 1 THEN GOTO 50
220 ? "HERE ARE ALL 20 NUMBERS IN PROPER ORDER"
230 FOR I = 1 TO 20
240 ? N(I);" ";
250 NEXT I
260 REM**LINE 210 OCCURS ONLY IF A SWITCH HAS BEEN MADE, THEN WE RE-CHECK
```

*Be sure the class knows why S does not go to 20 in line 60.

INSTRUCTIONAL OBJECTIVE

To introduce basic concepts of probability and develop some programs to illustrate these concepts.

PROCEDURES

Permutations

How many four-letter arrangements could we make by using the letters of the word TIME (using each letter once)?

TIME	TMEI	TEIM	TIEM	TMIE	TEMI
IMET	MEIT	EIMT	IEMT	MIET	EMIT
METI	EITM	IMTE	EMTI	IETM	MITE
ETIM	ITME	MTEI	MTIE	ETMI	ITEM

We can easily count 24 different arrangements or permutations. Each arrangement is different from the others. Order of arrangement makes a difference.

In general, to arrange n letters (all different) there are n choices for the first, n-1 for the second, n-2 for the third and so on. This kind of calculation is called factorial, written as N! 4!=(4)(3)(2)(1)= 24

A routine to compute factorials is:

```
10 INPUT "FIND THE FACTORIAL OF ";N
20 A = 1
30 FOR X = N TO 1 STEP -1
40 A = A * X
50 NEXT X
60 ? N;" FACTORIAL IS EQUAL TO ";A
```

Adjust INPUT so that only integers greater than one are allowed.

Suppose we wish to find the number of four-letter words formed from using letters of COMPUTE (without duplicating letters). Four positions must be filled. There are seven choices for the first, six for the second; there are five left for the third choice and four for the fourth. $7 \cdot 6 \cdot 5 \cdot 4$ or 840 different permutations. $7 \cdot 6 \cdot 5 \cdot 4 = \frac{7!}{3!}$ or $\frac{7!}{(7-4)!}$

7 possibilities
4 to be chosen

The permutations of N things taken R at a time = $_{N}P_{R} = \frac{N!}{(N-R)!}$

```
10 INPUT "TOTAL NUMBER OF DIFFERENT LETTERS ";N
20 INPUT "USING HOW MANY LETTERS? ";R
30 A = 1
40 FOR X = N TO N-R+1 STEP -1
50 A = A * X
60 NEXT X
70 ? "THE ANSWER IS ";A
```

Combinations

With permutations order makes a difference. This is not so with combinations. We may think of combinations as selection of items where order of selection is not important, as on a committee. The number of combinations formed from TIME, using four letters, is 1.

$$_{N}C_{R} = \frac{N!}{(N-R)!\ R!}$$

```
10 INPUT "TOTAL NUMBER OF ITEMS ";N
20 INPUT "HOW MANY AT A TIME?" ;R
30 REM**FIRST FIND N! / (N-R)!
40 A = 1
50 FOR X = N TO N-R+1 STEP -1
60 A = A * X
70 NEXT X
80 REM**DIVIDE BY R!
90 FOR X = R TO 1 STEP -1
100 A = A/X
110 NEXT X
120 ? "THE NUMBER OF COMBINATIONS IS ";A
```

What adjustment must be made to find permutations from letters not all of which are different?

Example: the word ALL can be written as ALL LAL LLA

Switching the two L's does not result in a noticeable difference. Taking 3! as the answer counts every distinguishable arrangement twice. Our answer should be $\frac{3!}{2!}$.

To arrange AAAABBBCCD, we have $\frac{10!}{4!\ 3!\ 2!}$

Write a program to arrange 20 students in a classroom.

Think about the number of ways four boys can be seated in a circle. The answer is not 4! but $\frac{4!}{4}$ or 3! since a circular arrangement has no beginning point. To arrange 10 keys on a key ring, $\frac{9!}{2}$. We divide by two since we can actually achieve different permutations by simply flipping the key chain over, so only half as many moves of keys are made. Write programs about hands dealt from a deck of 52 cards, about arranging books on a shelf or selecting people for a committee.

Probability

If an event can occur in T different ways, N of which we consider a success, we say the probability that we will have a success is N/T. In a group of 30 adults the probability that the first adult will be one of five committee members is

$$\frac{{}_{29}C_4}{{}_{30}C_5}$$

The numerator ${}_{29}C_4$ represents chosing the remaining four adults for the committee. It is the number of ways it can happen. ${}_{30}C_5$ is the number of ways any five can be chosen. Probability equals the number of ways something can happen (successes) divided by the number of ways anything can happen.

This program deals with the probability that two or more people in a group have the same birthday.

For five people the probability that all have different birthdays is

$P = \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdot \frac{361}{365}$ Therefore, the probability that at least two have the same birthday is 1-P. Here is a program to do the above.

```
 5 P = 1
10 INPUT "HOW MANY PEOPLE ARE IN THE GROUP" ;N
20 FOR X = 365 TO 365 - N + 1 STEP -1
30 P = P * X/365
40 NEXT X
50 ? "THE PROBABILITY TWO OR MORE PEOPLE IN A GROUP
   OF ";N;" HAVE THE SAME BIRTHDAY IS "; 1-P
```

To find the probability that 10 people have a different birth date than you, multiply the probability $\frac{364}{365}$ by 10 and subtract from 1.

Write the program.

INSTRUCTIONAL OBJECTIVE

To write a major program using the commands IF--THEN, FOR--NEXT, GOSUB, INPUT and PRINT in the form of a multiple choice quiz and print a final score.

PROCEDURES

This project involves many commands in a program designed as a multiple choice quiz. It can be altered to suit any subject area.

1. Ask the class to write five questions and three choices for each. The PRINT command and HTAB and VTAB are used.

```
100 HTAB4 : ?"1. WHAT IS THE CAPITAL OF THE U.S.?"
110 HTAB10 : ?"A. NEW YORK"
120 HTAB10 : ?"B. WASHINGTON"
130 HTAB10 : ?"C. BOSTON"
```

   Line 90 might be VTAB4.

2. Once the questions and choices have been determined, add an INPUT after each series.

```
140 INPUT A$
```

3. Now we'll evaluate each response for a correct or incorrect reply.

```
150 IF A$ = "B" THEN GOSUB 1000
160 IF A$ <> "B" THEN GOSUB 2000
```

   *Note the use of quotation marks. Students seem to forget them.

   In line 150 we handle a correct reply. The program will jump to line 1000 where it will PRINT a congratulatory statement and then RETURN. The next line it encounters after its return will be line 160. If 160 said GOSUB 2000 it would execute this command. Line 160 should be done only for incorrect replies so we state IF A$ DOES NOT EQUAL "B". The GOSUB routine located at 2000 will produce a negative statement.

4. The two GOSUBs must be developed now. The random generator function is used to chose a number 1 through 5. Depending on its result, a certain PRINT will occur.

```
1000 P = INT (5 * RND (1)) + 1
1010 IF P = 1 THEN ? "SUPER JOB!"
1020 IF P = 2 THEN ? "YOU ARE SO RIGHT"
1030 ↓
1060 RETURN
```

   OR

   use ON P GOTO 1010, 1020, 1030 .....

```
2000 Q = INT (5 * RND (1)) + 1
2010 ON Q GOTO 2020, 2030, 2040, 2050, 2060
2020 ? "NO, YOU MADE AN ERROR"; GOTO 2070
2030 ? "SORRY"; GOTO 2070
        ↓
2070 RETURN
```

OR

```
2020 ?"NO" : RETURN
2030 ? "SORRY":RETURN
etc.
```

5. Now introduce a counter so that a final grade can be determined. 10 T = 0 In the beginning of the program the counter should be set to zero in case it does not currently have this value.

   Line 150 IF A$ = "B" THEN T = T + 1 : GOSUB 1000

   Each time a correct response is made, T is incremented by 1. (Assuming that the last question ends before 900.)

```
900 ?"YOUR SCORE WAS "; T * 20
990 END
```

   Have students omit 990 and predict what would happen.

   The program could be expanded to PRINT another statement concerning the score.

```
920 IF T<3 THEN ?"STUDY MORE NEXT TIME"
930 IF T>2 AND T<5 THEN ?"NOT BAD"
940 IF T = 5 ?"TERRIFIC!"
```

   Doctor program by timers FOR I = 1 TO 200 : NEXT I and the HOME command so only one question appears at a time.

   Introduce program with a statement explaining that this is a multiple choice quiz.

NOTES

Students should be encouraged to plan for the unexpected. That is, what if someone INPUTs an E or Y? Modify the program to handle such replies.

INSTRUCTIONAL OBJECTIVE

To understand the READ--DATA command.

PROCEDURES

1.
```
10 DATA 42.6, 17, -3, 0, .4, 928
20 READ A
30 PRINT A
40 GOTO 20
```

The computer will ignore a DATA line (line 10) until it reaches a READ instruction. Line 20 tells the computer to READ one number from the DATA line. Commas must be used to separate variables in the DATA. No comma should occur at the end of the DATA, nor should it be used for numbers as 1,432 (one thousand four hundred thirty two). Line 30 commands a PRINT, line 40 sends the program back to line 20 where it encounters another READ. The computer "remembers" that it read 42.6 as the value of A on round one, and now choses 17 as the value for A. When the program is RUN an "OUT OF DATA" message occurs upon completion of the print-out. We will solve this problem later.

Change line 20 to READ A, B, C
Change line 30 to PRINT A, B, C
(or PRINT B or PRINT B,C)

This new READ line looks for DATA from line 10 in groups of three. What will happen if line 10 doesn't contain a multiple of three for the number of DATA items?

2. Explain that READ--DATA is a command like FOR--NEXT, IF--THEN, in that both words must occur in the program. The DATA statement can be listed anywhere within the program.

```
10 DATA SMITH, 90,80,100
20 DATA JONES, 40, 60, 80
30 DATA BROWN, 80, 70, 80
40 READ N$, A, B, C,
50 ?"THE AVERAGE FOR ";N$;" IS ";(A+B+C)/3
60 ?"THE GRADES WERE ";A;" ";B;" ";C
70 GOTO 40
80 END
```

When students RUN the above program, an error message of OUT OF DATA will occur. Add the following lines.

```
35 DATA FINISHED,0,0,0
45 IF N$ = "FINISHED" THEN END
```

Now use a FOR--NEXT along with READ--DATA to find five averages.

SUPPLEMENTARY ACTIVITIES

The READ--DATA command can be used in a variety of ways. Students could write a program that would check INPUT with the DATA, if a match is found then PRINT a statement....

Example: Animals/Animaux

which changes English words to their French counterpart.

```
 1 HOME
20 ?"*****************"
30 VTAB5 : HTAB4 : ?"UNICORN     BUTTERFLY     FOX"
40 VTAB7 : HTAB4 : ?"TURTLE      CAT           DOG"
50 VTAB9 : HTAB4 : ?"RABBIT      REINDEER      MONKEY"
60 VTAB11 : HTAB4 :?"HORSE       GUINIE PIG    BEAR"
70 VTAB13 : HTAB4 :?"KOALA       PANDA         ELEPHANT"
80 ?"*****************"
100 FOR X = 1 TO 1000 : NEXT X ; VTAB18 : ?"WHAT IS YOUR
    FAVORITE ANIMAL?"
110 INPUT R$
120 READ A$, B$
130 IF A$ = "END" THEN ?"SORRY,I DON"T KNOW THAT ANIMAL"
    : END
140 IF R$ = A$ THEN ?"YOUR FAVORITE ANIMAL IS A ";A$;
    " IN FRENCH IT IS ";B$ : END
150 GOTO 120
200 DATA UNICORN, LE LICORNE, RABBIT, LA LAPINE, CAT,
    LE CHAT, DOG, LE CHIEN,.....END, END
```

Line 140 could be changed to let the program find more than one match.

```
- 140 IF R$ = A$ THEN ?"YOUR FAVORITE ANIMAL IS A ";A$;" IN FRENCH IT IS "
    ;B$: RESTORE : GOTO 100
```

The command RESTORE returns the pointer to the beginning of the DATA statements. Replace the command END with RESTORE in line 130 and add GOTO 100.

NOTES

Other possible uses are for -

a) singular vs. plural of nouns
b) present vs. past tense of verbs
c) states vs. capitals

INSTRUCTIONAL OBJECTIVE

To write a program using the READ--DATA command that can be used for drill and practice.

PROCEDURES

The following program constitutes a spelling drill using READ--DATA.

First a word appears and the user must enter the same word. If he does so correctly the word flashes on the screen, then disappears and he must remember how to spell the word and enter it again. After successfully completing the above, the word flashes once again, but for a shorter duration. He must INPUT the same word one more time. If, at any point the INPUT does not match, the program returns to the beginning.

```
 10 HOME
 20 READ X$
 30 IF X$ = "QQQ" THEN GOTO 100
 40 HOME : VTAB(9) : HTAB(13) : ?X$
 50 VTAB(11) : HTAB(12) : INPUT Y$
 60 IF Y$ = X$ THEN GOSUB 200 : GOTO 20
 70 IF Y$ <> X$ THEN GOTO 40
100 HOME : HTAB(6) : VTAB(10) : ?"THAT'S IT
    FOR NOW FOLKS!" : END
200 SPEED = 50
210 HOME
220 HTAB(12) : VTAB(8) : ?X$
230 FOR X = 1 TO 1500 : NEXT
240 HOME
250 HTAB(11) : VTAB(10) : INPUT Y$
260 IF Y$ = X$ THEN HTAB(12) : VTAB(15) : ?"GREAT!"
270 IF Y$ <> X$ THEN 40
300 SPEED = 255
310 HOME : HTAB(12) : VTAB(10) : ?X$
```

```
320 FOR Z = 1 TO 400 : NEXT Z

330 HOME : HTAB(12) : VTAB(10) : INPUT Y$

340 IF Y$ = X$ THEN HTAB(12) : VTAB(15) :?"GOOD
    SPELLING!" : FOR Z = 1 TO 1000 : NEXT Z

350 IF Y$<>X$ THEN GOTO 40

360 RETURN

400 DATA "CRANE", "RECEIVE", "PRINCIPLE",....."QQQ"
```

SUPPLEMENTARY ACTIVITIES

Have the class write a program using READ--DATA for a speed quiz. The FOR--NEXT format should be used as a timer. It could easily be changed to include spellings of foreign words.

This program could also be changed to use as a simple speed reading drill. Either shorten the FOR--NEXT time interval or increase the length of the phrase that appears on the screen. Rather than re-typing the entire phrase list three choices two of which, although relevant to the phrase, are misinterpretations, along with the one correct response. Have the user select from A, B, or C the one which most closely identifies the presented phrase.

INSTRUCTIONAL OBJECTIVE

To learn how to re-arrange data by changing position within an array.

PROCEDURES

On the blackboard, write five numbers, not in order of magnitude.

Example: 10, 7, 20, 11, 3 --from a DATA line

Label these as A(1) = 10
A(2) = 7
A(3) = 20
A(4) = 11
A(5) = 3

} result after a FOR I = 1 TO 5
READ A(I)
NEXT I

Now compare the first pair (A(1) and A(2)). Since A(1) is greater than (10 and 7) A(2), we'll switch them. The original value of A(1) must be stored in another location, T. So T = A(1). Now let A(1) be replaced with A(2).

A(1) = A(2), so A(1) = 7 and A(2) is still 7

We must now make A(2) equal to 10, our T value. A(2) = T
Thus one switch has been made. Study the following program. When F remains zero, no switches were made; our order is perfect so the list is printed.

```
 10 DIM A$(20)
 20 FOR I = 1 TO 16 : READ A$(I) : NEXT I
 30 F = Ø : I = 1
 40 IF A$(I)< = A$(I + 1) THEN GOTO 90
 50 T$ = A$ (I + 1)
 60 A$ (I + 1) = A$(I)
 70 A$(I) = T$
 80 F = 1
 90 I = I + 1 : IF I< = 15 THEN GOTO 40
100 IF F = 1 THEN GOTO 30
110 FOR I = 1 TO 16
120 ? A$(I)
130 NEXT I
140 DATA SIXTEEN, FOUR, ONE, TEN, ELEVEN, THREE
150 DATA SEVEN, NINE, SIX, TWELVE, THIRTEEN, FIFTEEN
160 DATA TWO, FIVE, EIGHT, FOURTEEN
```

After the explanation at the board, these results may really surprise the class. Ask what happened; why? What easy changes could be made for numbers one through sixteen?

INSTRUCTIONAL OBJECTIVE

To be able to manipulate CHR$ within a program. Specifically, CHR$(8)-Backspace.

PROCEDURES

Students should be familiar with ASCII codes and CHR$. If not, do an introduction at the board and spend time at the computer doing things such as ?CHR$(81) and ?ASCII(A)

```
 10 B$ = CHR$(8) : REM**BACKSPACE
 15 HOME
 20 SPEED = 150 : HTAB6 : VTAB10:
 30 PRINT "I AM SO DESPERATE";
 40 FOR X = 1 TO 300 : NEXT X
 50 FOR L = 1 TO 17 : REM**NUMBER OF LETTERS AND SPACES
*60 PRINT B$;" ";B$;:REM*BACK UP, PRINT SPACE, BACK UP
 70 NEXT L
 80 FOR X = 1 TO 300 : NEXT X
 90 PRINT "I COULD EAT A LINE";
100 FOR X = 1 TO 300 : NEXT X
110 FOR L = 1 TO 18 : REM**NUMBER OF LETTERS AND SPACES
120 PRINT B$; " ";B$;
130 NEXT L
140 FOR X = 1 TO 300 : NEXT X
150 GOTO 15
```

Now experiment--

```
10 FOR X = 1 TO 100
20 ?CHR$(X)
30 NEXT X
```

Have students write programs to get different effects by using CHR$ and INPUTs for the X.

SUPPLEMENTARY ACTIVITIES

Write a program to identify the next sequential letter of the alphabet. Use a random generator to chose the first letter, print it on the screen and have the user type in the next letter. Make use of CHR$ and ASCII codes.

NOTES

*Line 60 - show this at the board.

Example: ABC□
B$ backs the cursor
AB□
" " prints a space
AB_□
B$ backs it again

AB□
and repeat again to erase the B.

INSTRUCTIONAL OBJECTIVE

Use programming to find roots of a polynomial.

PROCEDURES

Before writing a program which will divide a polynomial by a binomial in the form of X-D, review the process of synthetic division.

To divide $4x^3 + 2x^2 - 7x - 1$ by $x - 3$:

```
3 |  4   2   -7   -1
      12  42  105
    ------------------
      4  14   35  104
```

The result is $4x^2 + 14x + 35$ with a remainder of 104. Also note that
$F(3) = 4(3)^3 + 2(3)^2 - 7(3) - 1$
$= 108 + 18 - 21 - 1 = 104$ (our remainder)

```
              P(4) P(3) P(2)  P(1)
               ↓    ↓    ↓     ↓
          3 |  4    2   -7    -1
D  ↗            12   42   105
          ------------------------
               4   14   35    104
               ↗    ↑    ↑     ↖
            A(4)  A(3) A(2)    A(1)
          same as found         the remainder
            P(4)  by
                  adding P(3) + A(4)*D
```

Synthetic Division

```
 10 DATA 3
 20 REM**FIRST DATA WILL REPRESENT  OUR DIVISOR
 30 DATA 4, 2, -7, -1
 40 REM**SECOND DATA REPRESENTS COEFFICIENTS OF POLYNOMIAL
 50 REM**READ DIVISOR
 60 READ D
 70 REM**READ POLYNOMIAL COEFFICIENTS, THERE ARE 4 HERE
 80 FOR I = 4 TO 1 STEP -1
 90 READ P(I)
100 PRINT P(I);" ";
110 NEXT I
120 PRINT " IS TO BE DIVIDED BY X- ";D
130 A(4) = P(4)
140 REM**A(4) IS FIRST COEFFICIENT OF ANSWER, ALWAYS
    EQUAL TO THE FIRST COEFFICIENT
150 PRINT "THE RESULT IS "; A(4);
160 FOR T = 3 TO 1 STEP -1
```

```
170 A(T) = P(T) + A(T + 1) * D
180 PRINT " "; A(T);
190 NEXT T
195 PRINT
200 IF A(1) = 0 THEN PRINT "F(";D;") IS A ROOT OF THE
    POLYNOMIAL"
210 IF A(1)<>0 THEN PRINT "F(";D;") = ";A(1)
```

Change the program so that INPUT statements are used instead of READ--DATA. Remember to change lines 80 and 160.

The next program finds all roots of a polynomial which are integers. If the polynomial has any roots which are integers, they must be factors of the constant term in the original polynomial. The remainder in the previous program would equal zero when a root occurs.

This first program for finding integral roots will work for a fourth degree polynomial defined as F(X). It also uses a READ--DATA technique.

Integral Roots

```
  5 DATA 1, -2, 3, 10, -7
  7 K = Ø
 10 Z = P(5) * X^4 + P(4) * X^3 + P(3) * X^2 + P(2)
    * X + P(1)
 20 FOR I = 5 TO 1 STEP -1
 30 READ P(I)
 40 PRINT P(I);" ";
 50 NEXT I
 60 REM **LINES 30 TO 50 READ THE FIVE POLYNOMIAL
    COEFFICIENTS
 70 FOR X = -P(1) TO P(1) STEP SGN (P(1))
 80 REM**WE WANT TO TEST NEG AND POSITIVE NUMBERS
    THROUGH P(1)
 90 REM**IF P(1) IS NEGATIVE WE MUST USE A NEGATIVE STEP
    (EXAMPLE P(1) = -3, FOR X = -3 TO 3 STEP -1)
100 IF X = 0 THEN GOTO 200 : REM**CAN'T DIVIDE BY 0
110 IF P(1) / X < > INT (P(1)) / X) THEN GOTO 200
120 REM**DOESN"T DIVIDE EVENLY, CAN'T BE FACTOR OF
    P(1), BUT IS NOT A ROOT SO IT CAN'T BE AN INTEGRAL ROOT.
130 IF Z < > Ø THEN GOTO 200
140 REM*X DIVIDED EVENLY, IS A FACTOR OF P(1), BUT IS
    NOT A ROOT
150 PRINT "ONE ROOT IS "; X
160 IF K = 1 : REM**AT LEAST ONE ROOT WAS FOUND
200 NEXT X
250 IF K = Ø THEN PRINT "SORRY, NO ROOTS WERE FOUND"
300 END
```

Improve the program so it can be used for any polynomial.
For a polynomial of degree N

F(X) would equal P(N + 1)* X↑N + P(N)* X↑(N-1) +....P(1)

The value of F(X) is found by taking the sum of (N+1) terms

F(X) for a 4th degree polynomial =

$P(5)X^4 + P(4)X^3 + P(3)X^2 + P(2)X + P(1)$

Use this FOR--NEXT loop. After line 110, use GOSUB 400

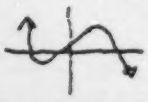
```
400 P = 0
410 FOR T = N TO 1 STEP -1
420 P = P + P(T) * X↑(T-1)
430 NEXT T
440 RETURN
```

We must also INPUT N where N stands for the number of terms (five terms for a 4th degree polynomial). There is no need for line 5 or 10. Add an INPUT line -- 15 INPUT "HOW MANY TERMS?" ;N

```
Line 20 now is FOR I = N TO 1 STEP -1
     30        INPUT P(I)
     40        ? P(I)
     50        NEXT I
```

(For polynomials of large degree use a DIM statement.)

Not all real roots of a polynomial are integers. To find others we shall use the following process. Notice, then, on this graph there are three roots. Values of F(X) near each root have opposite signs. That is F(X) may have positive values, then becomes zero, then has negative values.



If two points are on opposite sides of the x-axis, then the graph must cross the axis between those two points. That is, there must by a root between the two x values.

Here is the beginning of a program to find all real zeros of a polynomial. It will be improved shortly since it now lists intervals within which the root falls, of width one.

```
 5 T = 0
 8 FOR X = -10 TO 10
10 REM**DEFINE THE FUNCTION
20 A = 3 * (X)^3 + 2 * (X)^2 - 7 * (X) -4
30 B = 3 * (X + 1)^3 + 2 * (X + 1)^2 - 7 * (X + 1) - 4
60 IF A * B  0 THEN GOTO 80
70 PRINT "THERE IS A ROOT BETWEEN  ";X; " AND "; X + 1
75 T = 1
80 NEXT X
90 IF T = 0 THEN PRINT "SORRY, NO ROOTS HAVE BEEN FOUND"
```

Add a line to cover the possibility that A, B, or both are zero. What does that mean?

This program allows the user to INPUT the values in the FOR--NEXT loop and STEP value.

```
 5 T = 0
10 REM**ALLOW FOR INPUT OF STEP VALUE AND LIMITS OF SEARCH
20 INPUT "THE LEFT ENDPOINT OF OUR SEARCH WILL BE "; L
30 INPUT "THE RIGHT ENDPOINT WILL BE ";R
40 INPUT "THE INCREMENTS WILL BE ";S
50 IF S = 0 THEN END
60 FOR X = L TO R STEP S
70 A = 4 * X^3+ 2 * X^2 - 11 * X - 2 : REM**DEFINE
   FUNCTION
80 B = 4 * (X + 1)^3 + 2 * (X + 1)^2 - 11 * (X + 1) - 2
90 IF A * B> 0 THEN GOTO 110
100 PRINT "THERE IS A ROOT BETWEEN ";X;" AND ";X + S
105 T = 1
110 NEXT X
120 IF T = 0 THEN PRINT "SORRY, NO ROOTS HAVE BEEN FOUND"
130 INPUT "DO YOU WISH TO TRY AGAIN?"; R$
140 IF R$ = "Y" OR R$ = "YES" THEN GOTO 5
```

Check for possible errors occuring due to INPUTs. That is, what if L>R? What should happen if A*B = Ø? If no change in sign occurred, several things could have happened. We could be right in assuming no roots are located within the interval. However, our increment might be too large and as a result two zeros lie within the same interval. We must also try to get our limits large enough to include roots.

INSTRUCTIONAL OBJECTIVE

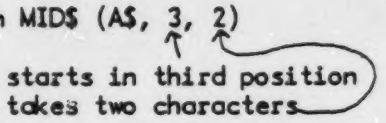
To use LEFT, RIGHT and MID$ in programs.

PROCEDURES

Begin with simple PRINT statements -

```
10 A$ = "ABCDEF"
20 ? LEFT$ (A$,2)
```

Change line 20 to RIGHT$ (A$,4) then MID$ (A$, 3, 2)



Work up to FOR--NEXT and LENGTH of a string.

```
 5 A$ = "ABCDEF"
10 X = LEN(A$)
20 FOR I = 1 TO X
25 ? LEFT$ (A$, I)
30 NEXT I
```

See if anyone in the class can guess the correct outcome. Change line 25 by using MID$ and try to get one letter at a time A through F.

Since everyone enjoys seeing their name in print, try something like this.

```
10 INPUT "WHAT IS YOUR NAME?"; N$
20 FOR X = 1 TO LEN(N$)
30 ? LEFT$ (N$, X)
40 NEXT X
```

See if students can make their name gradually disappear, i.e. starting with a program in which their entire name appears, then one letter drops off repeatedly.

Can they write a program using MID$ to show their names starting in the middle (more or less) and adding one letter to each side.

Example: Susan Smith

```
-
N-S
AN-SM
SAN-SMI
USAN-SMIT
SUSAN-SMITH
```

ACTIVITY/WORKSHEET

```
10 HOME

20 ? "TYPE YOUR NAME FOR ME AND THEN I WILL TELL YOU WHICH
   POSITION ANY LETTER IS IN."

30 INPUT N$

40 ? "NOW ENTER A NUMBER BETWEEN 1 AND "; LEN(N$);" AND I WILL NAME
   THE LETTER IN THAT POSITION."

50 INPUT N

60 IF N < 1 OR N > LEN(N$) THEN ? "FOLLOW DIRECTIONS" : GOTO 40

70 ? MID$ (N$, N, 1); " IS THE LETTER IN POSITION NUMBER ";N;
   " IN YOUR NAME."
```

Have students try the reverse which poses more problems. That is, enter a letter and say which position it is in.

```
Enter L$
FOR I = 1 TO LEN (N$)
IF MID$ (N$, I, 1) = L$ THEN ? ----------------
NEXT I
```

Of course, letters often occur in multiples and each should be recorded.

INSTRUCTIONAL OBJECTIVE

To make use of concatenation in a word scramble program.

PROCEDURES

```
 5 INPUT "GIVE ME A WORD AND I'LL ATTEMPT TO SCRAMBLE IT FOR
   YOU"; OW$

10 REM**OW STANDS FOR ORIGINAL WORD

20 SW$ = OW$ : REM**SW MEANS SAVED OR STORED WORD SO A COMPARISON
   CAN LATER BE MADE

30 NW$ = " ": REM**NW IS OUR NEW SCRAMBLED WORD

40 L = LEN(OW$) : REM**L IS LENGTH OF ORIGINAL WORD

50 FOR X = 1 TO L

60 T = INT (RND(1) * LEN(OW$))+ 1 : REM**T IS THE POSITION OF
   LETTER WE WILL BE TAKING

70 NW$ = NW$ + MID$(OW$, T, 1) : REM**WE ARE BUILDING A NEW WORD
   BY TAKING 1 LETTER AT A TIME FROM THE ORIGINAL WORD AND ADDING
   IT TO NW

80 ?"THE RANDOM NUMBER ";T;" WAS CHOSEN THIS TIME AND THE NEW WORD,
   AT THIS POINT IS "; NW$

90 OW$ = MID$ (OW$, 1, (T-1) + MID$ (OW$, (T+1), L) : REM**THE
   ORIGINAL WORD NOW HAS THE CHOSEN LETTER DELETED.  IT CONSISTS OF
   ALL LETTERS BEFORE IT AND ALL LETTERS AND SOME SPACES (IRRELEVANT)
   AFTER IT.

100 ? "THE ORIGINAL WORD IS NOW "; OW$

110 NEXT X

120 IF NW$ = SW$ THEN ? "OOPS!!  THE SCRAMBLED WORD AND THE NEW WORD
   TURNED OUT TO BE IDENTICAL.  I'LL TRY AGAIN." : GOTO 30

130 ?"THE SCRAMBLED WORD IS     "; NW$
```

INSTRUCTIONAL OBJECTIVE

To understand the difference between sequential and random access files and write programs using each.

To use MON as a visual instruction device and appreciate its potential in de-bugging programs.

PROCEDURES

Explain the difference between sequential files and random access files as far as method of saving data is concerned.

Work with this program or one similar.

```
10 D$ = CHR$ (4)
20 ? D$ "OPEN CABINET"
30 ? D$ "WRITE CABINET"
40 ? "MEESE"
50 ? "SHULTZ"
60 ? "WEINBERG"
70 ? "WATT"
80 ? D$ "CLOSE CABINET"
```

Explain that hitting the return key after 40, 50, 60, 70--signals the computer that a record is complete. The ASCII code for a return character is CHR$(141).

Students should be aware of a WRITE--PRINT combination for making the file and a READ--INPUT combination for retrieving the file.

D$ = CHR$(4) can also be written as D$ = " " where a CTRL D comes between the quotation marks. This signals the computer that commands to the drive are forth coming.

Poll the class for ideas on the output of a RUN, then RUN the program. Most students are very surprised at the lack of video action, but do notice the red light on for the disk drive.

Type CATALOG. Notice the T CABINET on the listing. Mention the differences between A, I, T and B if this hasn't been done.

Now type MON C,I,O which monitors the commands, input and output from disk. Run the program.

Try typing MON I versus MON C, O etc. Now try NOMON C, I, O which cancels the MON command changing from MON C, I, O to MONO does no good.

*Stress closing any file after opening it or else weird things can go on!

Save the program listed above as SAVE MAKE CABINET.

Now write another program to get the file called CABINET

```
10 D$ = CHR$(4)
20 ?D$ "OPEN CABINET"
30 ?D$ "READ CABINET"
40 INPUT A$
50 INPUT B$
60 INPUT C$
70 INPUT E$
80 ?D$ "CLOSE CABINET"
```

Run the above. Save it as RETRIEVE CABINET. Add line 75 INPUT F$. Try to predict the outcome. The disk has no more information so an OUT OF DATA error occurs. Delete 75. Change 70 to INPUT D$. Now RUN the program. Line 80 produces WATT CLOSE CABINET. D$ has been assigned CHR$(4) in line 10 and should not be used in any other connection.

Now write a program to illustrate random access files.

```
 5 I = 1
10 D$ = CHR$(4)
20 ?D$ "OPEN DOORS, L40"
30 ?D$ "WRITE DOORS, R"; I
40 ?"RAISED PANEL"
50 ?D$ "CLOSE DOORS"
```

Line 20 specifies the length of each record as 40; whether or not all 40 character spaces are used, the computer sets aside the space.

When RUN, the program creates a text file labeled DOORS. It contains one record RAISED PANEL record #1. Change 5 to I = 6. RUN. Now there is one record, #6

SAVE the above as MAKE DOORS.

Now write RETRIEVE DOORS.

```
10 D$ = CHR$(4)
20 ? D$ "OPEN DOORS, L40"
30 ? D$ "READ DOORS, R"; I
40 INPUT A$ (I)
45 ? A$(I)
50 ? D$ "CLOSE DOORS"
```

Insert 5 I = 1 and RUN. Students may be surprised at the result. Ask how line 5 should be changed. If 5 I = 6, results are as expected. Line 20 must specify the correct length or the pointer will not read from the beginning of each record, but will overlap and cause problems.

Now write a program to create a random access file which has 10 records.

```
** 5 DIM A$(10)
  10 FOR I = 1 TO 10
  20 INPUT "NAME A TYPE OF DOOR"; A$(I)
  30 NEXT I
  40 D$ = CHR$(4)
  50 ?D$ OPEN DOORS, L40"
  60 FOR I = 1 TO 10
  70 ?D$ "WRITE DOORS, R"; I
  80 ? A$(I)
  90 NEXT I
 100? D$ "CLOSE DOORS"
```

Lines 10 - 30 allow the user to enter 10 types of doors (raised panel, screen, warped, garage, slammed, barn .....). They are stored in a one dimensional array labeled A$. Then lines 60 - 90 writes 10 records . **Don't forget a DIM statement if you work with more than 10 items in an array. RETRIEVE DOORS would be as follows:

```
10 D$ = CHRS(4)
15 ? D$ "OPEN DOORS, L40"
20 FOR I = 1 TO 10
30 ?D$ "READ DOORS, R"; I
40 INPUT A$(I)
50 NEXT I
60 ?D$ "CLOSE DOORS"
```

Not all 10 files must be read. Change line 20 and experiment.

If NOMON C,I,O is in effect, nothing will appear on the screen. Add lines -

```
70 FOR I = 1 TO 10
80 ?A$(I)
90 NEXT I
```

This program begins to compare records in a random access file.

MAKE INFORMATION

```
 10 FOR I = 1 TO 10
 20 INPUT "WHAT IS YOUR NAME? LAST NAME FIRST PLEASE.";
    N$(I)
 30 INPUT "WHERE DO YOU LIVE"; T$(I)
 40 INPUT "HOW MANY PEOPLE ARE IN YOUR IMMEDIATE FAMILY";
    M(I)
 50 NEXT I
100 D$ = CHR$(4)
110 ?D$ "OPEN INFORMATION, L100"
120 FOR I = 1 TO 10
130 ?D$ "WRITE INFORMATION, R"; I
140 ?N$(I)
150 ?T$(I)
160 ?M(I)
170 NEXT I
```

RETRIEVE INFORMATION

```
10 D$ = CHR$(4)
20 ? D$ "OPEN INFORMATION, L100"
30 FOR I = 1 TO 10 : ?D$"READ INFORMATION, R"; I
40 INPUT N$(I)
50 INPUT T$(I)
60 INPUT M(I)
70 NEXT I
80 ?D$ "CLOSE INFORMATION"
```

To get a list of names who have the greatest number of family members, add the following:

```
100 Z = 0
110 FOR I = 1 TO 10
120 IF M(I) > Z THEN Z = M(I)          Finds the greatest M
130 NEXT I
```

```
140 FOR I = 1 TO 10
150 IF M(I) = Z THEN PRINT "N$(I); " HAS "; M(I);
    "PEOPLE IN THE FAMILY"
160 NEXT I
```

Prints the list.

NOTES

In explaining problems caused by incorrect record length -- try this example.

The Mad Hatter commands "Move down, move down". Assuming chairs are equally spaced, an incorrect space move would result in an unexpected landing. Or imagine a ribbon cutting machine to cut off the letters of your school's name. If the machine was not correctly set to the proper length, your school name would be "unusual".

INSTRUCTIONAL OBJECTIVES

To use knowledge of text files in a practical, useful context.

PROCEDURES

Have the class spend some time thinking of suggestions for a text file program that they could use as a class.

One group decided to write a program to control bookkeeping of their class dues. Here is the beginning of their venture.

MAKE CLASS DUES

Puts ten names into active memory

```
10 FOR I = 1 TO 10 (number of students in class)
20 INPUT "NAME     "; A$(I)
30 INPUT "AMOUNT OWED  "; O(I)
40 NEXT I
```

creates a file

```
100 D$ = CHR$(4)
110 ?D$ "OPEN CLASS DUES, L50"
120 ?D$ "WRITE CLASS DUES, R"; I
130 ?A$(I)
140 ? O(I)
150 NEXT I
160 ?D$ "CLOSE CLASS DUES"
```

Each of the ten records consists of a name and an amount.

RETRIEVE CLASS DUES

Pulls records from disk to active memory

```
10 D$ = CHR$(4)
20?D$ "OPEN CLASS DUES, L50"
```

Reads ten records

```
30 FOR I = 1 TO 10
40 ?D$"READ CLASS DUES, R"; I
50 INPUT A$(I)
60 INPUT O(I)
65 NEXT I
70 ?D$ "CLOSE CLASS DUES"
```

Each record as a name ($) and amount

```
100 INPUT "WHO IS PAYING DUES?"; N$
110 INPUT "HOW MUCH DO YOU WISH TO PAY"; X
```

Searches for a name match

```
200 FOR I = 1 TO 10
210 IF N$ = A$(I) THEN GOTO 400
220 NEXT I
300 ?"SORRY" : END
```

Match found, change O(I)

```
400 O(I) = O(I) - X
```

Re-write one record, need name and new amount.

```
500 ?D$ "OPEN CLASS DUES, L50"
510 ?D$ "WRITE CLASS DUES, R "; I
520 ?A$(I)
530 ?O(I)
540 ?D$ "CLOSE CLASS DUES"
```

There are several questions which usually arise as to efficiency. Lines 520 and 530 re-write not only the corrected amount of dues now owed, but also re-write the name of the student. Line 210 checks each entire name for a match; very time consuming. If anyone INPUTs DAVE instead of DAVID, no match will be found. We are only correcting one file at a time.

The most pressing problem was that of incorrectly spelling a name and not finding a match; or an INPUT of SMITH DAVE instead of SMITH DAVID as registered in the file.

To alter the program add:

```
230 ?"RE-ENTER PLEASE, NO SUCH NAME WAS FOUND" : GOTO 100
```

After a RUN, the lack of a solution being so simple was realized. The user could be in an endless loop, never correctly entering a name.

The class adjusted by adding the following:

```
 99 Q = Q
 ↓
225 Q = Q + 1
230 If Q< 2 THEN?"RE-ENTER PLEASE, NO SUCH NAME WAS FOUND ": GOTO 100
240 END
```

This solved the endless loop situation, but it was still obviously deficient. How could the user know the correct INPUT response?

The following replaced 240.

```
240?"THESE ARE ALL OF THE FILE NAMES THAT BEGIN WITH "; LEFT$(N$,1)
250 FOR I = 1 TO 10 (length of class)
260 IF LEFT$(A$, 1) = LEFT$(N$, 1) THEN ? A$
270 NEXT I
280 GOTO 99
```

Students also worried about an INPUT in line 60 -- could it be 225 or must it be 2.25. The class decided on the latter. The computer would only accept an INPUT with a decimal. Their solution was:

```
60 INPUT"PLEASE ENTER AMOUNT TO BE PAYED AS XX.XX"; O$(I)
62 IF RIGHT $(O$(I),3)<> "."THEN? "TRY AGAIN PLEASE" :
   GOTO 60
63 O(I) = VAL (O$(I))
```

NOTES

There always are over-ambitious students who imagine every conceivable problem with a program and get bogged down in complications. The easiest approach is to begin with the skeleton of a program, assume all is right, (no one will misspell etc.) then work on ironing out "what-if" situations later.

INSTRUCTIONAL OBJECTIVE

Realize there is more than one way to reach an objective by using files. The following section approaches one problem from two different views. The first makes use of two separate files, one entitled NAMES AND COURSES, the other GRADES, then combines them both in a RETRIEVE NAMES, AND COURSES, AND GRADES program. The second method creates one file, NAMES COURSES GRADES and uses it in the retrieve program.

PROCEDURES

By using arrays, we create two files. The first is a two dimensional array called NAMES AND COURSES.

RUN with MON C,I,O in effect.

MAKE NAMES AND COURSES

```
  5 NS = 11 : CO = 6 : REM**NS = NO. OF STUDENTS,
    CO = NUMBER OF COURSES
 10 DIM A$ (11,6)
 20 FOR A = 1 TO NS
 30 INPUT "NAME     :"; A$(A,0)
 40 FOR B = 1 TO CO
 50 INPUT "COURSE : ";A$(A,B)
 60 NEXT B
 70 NEXT A
100 D$ = CHR$ (4)
110 ?D$ "OPEN NAMES AND COURSES, L 200"
**120 FOR A = 1 TO NS
130 ? D$"WRITE NAMES AND COURSES, R"; A
140 FOR B = 0 TO CO
150 ?A$(A,B)
160 NEXT B
170 NEXT A
180 ?D$ "CLOSE NAMES AND COURSES"
```

**This must be before 130 -- want to write all of record 1 before we change to record 2.

MAKE GRADES

```
 10 D$ = CHR$(4)
 20 NS = 11 : CO=6: GD = 2 : REM**ONLY TWO GRADES FOR
    ILLUSTRATION
 30 ?D$ "OPEN GRADES, L198"
 40 FOR N = 1 TO NS
 50 ?D$ "WRITE GRADES, R "; N
 60 FOR S = 1 to CO
 70 FOR G = 1 TO GD
 80 ? - 1
 90 NEXT G                          (This file makes grades.)
100 NEXT S
110 NEXT N
120 ?D$ "CLOSE GRADES"
```

In MAKE NAMES AND COURSES, Lines 20 - 70 fill active memory with 11 student names, each with 6 courses. Each A$(number, 0) represents a name, associated with each number is A$ (same first number, changing second number). A$ (2,3) represents the third course chosen by the second student.

Lines 100 - 180 create the text file with 11 records each containing 1 student name and 6 courses. If someone has fewer than six courses, enter the letter F for free period.

NAMES AND COURSES is a two dimensional array with 11 rows, 7 columns. Names of students are entered, one per row, column number zero. Six courses are allowed for each student, entered in the appropriate row, columns 1 - 6.

Example:

COLUMN	0	1	2	3	4	5	6
ROW 1	SMITH	ENGLISH	MATH	HISTORY	BIOLOGY	MUSIC	ART
ROW 2	JONES	ENGLISH	MATH	SPANISH	PSYCHOLOGY	GYM	HOME EC

In MAKE GRADES, we have 11 records. Each record has twelve -1's.

This program allows us to change a grade for one of the six subjects. It uses <u>both</u> of the previously created files.

```
 5 NS = 11 : CO = 6 : GD = 2
10 DIM A$(11,6), B% (11,6,2)
20 D$ = CHR$(4)
30 ?D$ "OPEN NAMES AND COURSES, L200"
40  FOR A = 1 TO NS
50 ?D$ "READ NAMES AND COURSES, R "; A
60 FOR B = 0 TO CO
70 INPUT A$(A,B)
80 NEXT B
90 NEXT A
100 ?D$"CLOSE NAMES AND COURSES"
200 INPUT "NAME ; "; N$
210 INPUT "SUBJECT : "; S$
220 INPUT "GRADE : "; GA
230 FOR N = 1 TO NS
240 IF N$ = A$ (N,0) THEN GOTO 260
250 NEXT : ?"SORRY, NO NAME MATCHES.  TRY AGAIN." : GOTO 200
260 FOR S = 1 TO CO
270 IF A$ (N,S) = S$ THEN GOTO 290
280 NEXT : GOTO 200
290 ?D$ "OPEN GRADES, L198"
300 ?D$ "READ GRADES, R"; N
310 FOR A = 1 TO CO
320 FOR G = 1 to GD
330 INPUT B% (N,A,G)
340 NEXT G
350 NEXT A
360 FOR G = 1 to GD
370 IF B% (N,S,G) = -1 THEN GOTO 390
380 NEXT G : ?"SORRY, NO MORE ROOM" : END
390 B% (N,S,G) = GA
400 ?D$ "WRITE GRADES, R" ;N
410 FOR S = 1 TO CO
420 FOR G = 1 to GD
430 ? B% (N,S,G)
440 NEXT G
450 NEXT S
460 ?D$ "CLOSE GRADES"
```

Notice that B ranges 0--CO, since we want names as well as courses.

A$(1,0) first name
A$(1,1) his first course

Change variable to A to preserve S

Need to write 1 whole record --- can't just use 420

Now write a program without using MON C,I,O that will read all NAMES AND COURSES then read GRADES and print a chart showing each name, courses and grades for each.

This program does the same thing as the two previous programs, but only 1 file is used. Names, courses and grades are all on 1 file.

```
 5 NS = 2 : CO = 2 : GD = 3 : REM**2 NAMES, 2 COURSES, 3 GRADES FOR EACH.
10 DIM A$(2,2,3)
20 FOR A = 1 TO NS
25 INPUT "NAME :" ; A$(A,Ø,Ø)
30 FOR B = 1 TO CO
40 INPUT "COURSE :     "; A$(A,B,Ø)
45 FOR C = 1 TO GD
50 INPUT "GRADE  : "; A$(A,B,C)        ---- line 50 makes initial grades
                                            Ø or change line 320 in the
60 NEXT C                                   following program.
70 NEXT B
80 NEXT A
100 D$ = CHR$(4)
110 ?D$"OPEN NAMES COURSES GRADES, L200"
120 FOR A = 1 TO NS
130 ?D$ "WRITE NAMES COURSES GRADES, R " ;A
140 ?A$(A,0,0)
150 FOR B = 1 TO CO
160 ?A$(A,B,0)
170 FOR C = 1 TO GD
180 ?A$(A,B,C)
190  NEXT C
200  NEXT B
210  NEXT A
220 ?D$ "CLOSE NAMES COURSES GRADES"
```

This program allows a user to change a grade or retrieve names, courses and grades.

```
 5 NS = 2 : CO = 2 : GD = 3
10 D$ = CHR$(4)
20 ?D$ "OPEN NAMES COURSES GRADES, L200"
30 FOR A = 1 TO NS
40 ?D$ "READ NAMES COUSES GRADES, R "; A
50 INPUT A$(A,Ø,Ø)
60 FOR B = 1 TO CO
70 INPUT A$(A,B,Ø)
80 FOR C = 1 TO GD
90 INPUT A$(A,B,C)
100 NEXT C
110 NEXT B
120 NEXT A
130?D$ "CLOSE NAMES COURSES GRADES"
200 INPUT "NAME : ";N$
210 INPUT "SUBJECT : "; S$
220 FOR A = 1 to NS
230 IF N$ = A$(A,Ø,Ø) THEN GOTO 260
240 NEXT A
250 END
260 FOR S = 1 TO CO
270 IF S$ = A$ (A,S,0) THEN GOTO 300
280 NEXT S
290 END
300 INPUT "GRADE IS "; GA
310 FOR C = 1 TO GD
320 IF VAL(A$(A,S,C))=0 THEN GOTO 400 (-need VAL since original
330 NEXT C                              file had 0 entered as a
340 END                                 string)
400 A$(A,S,C) = STR$(GA)
410 ?D$ "OPEN NAMES COURSES GRADES, L200"
420 ?D$ "WRITE NAMES COURSES GRADES, R ";A
430 ?A$ (A,Ø,Ø)
440 FOR S = 1 TO CO
450 ?A$(A,S,Ø)
460 FOR C = 1 TO GD
470 ?A$ (A,S,C)
480 NEXT C
490 NEXT S
500 ?D$"CLOSE NAMES COURSES GRADES"
```

400 changes grade to a string before entering in file.

RUN all of these programs with MON C,I,O.

Write one program to list names, grades and courses for the method of using one file to enter information.

Stress to students that it is always easier to write a retrieve information file if you look at the MAKE information file as you work.

INSTRUCTIONAL OBJECTIVE

To use text files and write a library program.

PROCEDURES

Two programs follow. The first is entitled MAKE LIBRARY FILE and it allows five people to use the library, each of them can check out four books. The second program, RETRIEVE LIBRARY FILE, allows for addition or deletion to the original list of five names and a "1" for each initial book. Have the class try to add a section which determines if a fine should be collected and if so, how much (no easy task)!

MAKE LIBRARY FILE

```
 1 DIM N$(30)
 2 X$ = CHR$(47) : REM /
 5 I = 5
 6 INPUT "NAME : "; N$(I)
 7 IF I < 25 THEN I = I + 5 : GOTO 6
10 REM : MAKE LIBRARY FILE
12 D$ = CHR$(4)
14 ?D$; "OPEN LIBRARY FILE, L30"
16 ?D$; "DELETE LIBRARY FILE"    (in case there is one already)
30 ?D$; "OPEN LIBRARY FILE, L30"
45 FOR I = 5 TO 29
50 ?D$; "WRITE LIBRARY FILE, R "; I
60 ?X$
65 NEXT I
80 I = 5
90 ?D$ "WRITE LIBRARY FILE, R ";I
100 ? N$(I)
110 IF I < 26 THEN I = I + 5 : GOTO 90
120 ?D$ "CLOSE LIBRARY FILE"
```

Files records 5 to 29 with a /

Writes records 5, 10, 15, 20, 25 -- each with a name

We now have a file consisting of five names, each followed by four /. A / will later signify an empty space where a book name can be entered.

RETRIEVE LIBRARY FILE

```
110 DIM N$(40)
120 D$ = CHR$(4)
130 PRINT D$: "OPEN LIBRARY FILE, L30"
140 I = 5
150 PRINT D$; "READ LIBRARY FILE, R"; I
160 INPUT N$(I)
170 IF I< 25 THEN I = 1 + 5 : GOTO 150
180 PRINT D$ "CLOSE LIBRARY FILE"
190 GOTO 5000
340 REM : ADD A BOOK TO THE LIST
345 TEXT : HOME: VTAB(10) : HTAB(6)
350 I = I + 1
360 X = 0
370 PRINT "WHAT IS THE NAME OF THE BOOK?"
375 PRINT : PRINT: HTAB(6) : INPUT B$
376 PRINT : PRINT
380 INPUT "WHAT IS THE DATE DUE? --/--/--";R$
390 IF LEN (B$)> 20 THEN B$ = LEFT$ (B$,20)
400 B$ = B$ + R$
410 RESTORE
420 PRINT D$ "OPEN LIBRARY FILE, L30"
430 PRINT D$ "READ LIBRARY FILE, R"; I
440 INPUT X$
450 IF X$ = "/" THEN GOTO 490
460 X = X + 1
470 I = I + 1 : IF X< 4 THEN GOTO 430
475 GOTO 545
490 X$ = B$
500 PRINT D$;"WRITE LIBRARY FILE, R"; I
510 PRINT X$
520 PRINT D$;"CLOSE LIBRARY FILE"
525 TEXT
530 GOTO 5025
545 PRINT D$ "CLOSE LIBRARY FILE"
550 INPUT "DO YOU WISH TO TRY AGAIN?"; Y$
555 IF Y$ = "Y" OR Y$ = "YES" THEN GOTO 5025
560 REM : DELETE A BOOK FROM THE LIST
565 VTAB(4) : HTAB(5)
570 I = I + 1
580 T = 0
590 PRINT "WHAT IS THE NAME OF THE BOOK?"
595 ?: HTAB(6) : INPUT B$
600 IF LEN (B$)  20 THEN B$ = LEFT$ (B$,20)
610 PRINT D$; "OPEN LIBRARY FILE, L30"
620 PRINT D$; "READ LIBRARY FILE, R"; I
630 INPUT X$
670 M = LEN (X$)
675 IF X$ = "/" THEN GOTO 710
680 X$ = LEFT$ (X$, M - 8)
700 IF X$ = B$ THEN GOTO 730
710 T = T + 1: IF T  4 THEN I = I + 1 : GOTO 620
```

```
715 PRINT "SORRY, THERE IS NO SUCH TITLE ON FILE" : FOR Q = 1TO 2000:
    NEXT Q
717 PRINT D$ "CLOSE LIBRARY FILE"
718 GOTO 5060
730 X$ = CHR$ (47)
740 PRINT D$; "WRITE LIBRARY FILE, R; I
750 PRINT X$
760 PRINT D$; "CLOSE LIBRARY FILE"
765 GOTO 5025
850 I = I + 1
860 X = 0
865 POKE 32,5: VTAB(10) : PRINT
870 PRINT D$ "OPEN LIBRARY FILE, L30"
880 PRINT D$"READ LIBRARY FILE, R"; I
890 INPUT B$
900 PRINT B$
910 X = X + 1
920 IF X< 4 THEN I = I + 1 : GOTO 880
930 PRINT D$ "CLOSE LIBRARY FILE"
934 PRINT : PRINT
935 PRINT "HIT ANY KEY WHEN READY TO PROCEED."
936 TEXT : GET P$
937 ON VAL (P$) GOTO 5040
5000 REM :
5010 REM :      MAIN MENU
5020 REM :
5025 HOME
5030 POKE 32,5 : VTAB(4)
5035 PRINT
5040 REM: WHOSE'S FILE DO YOU WISH TO ACCESS?
5050 RESTORE
5060 PRINT "WHOSE FILE DO YOU WISH"
5062 INPUT "TO ACCESS?"; Q$
5065 PRINT : PRINT
5070 I = 5
5080 IF N$(I) = Q$ THEN GOTO 5110
5090 IF I < 25 THEN I = I + 5 : GOTO 5080
5100 GOTO 5060
5110 PRINT
6000 PRINT "THIS PROGRAM WILL ALLOW YOU"
6010 PRINT "TO:"
6020 POKE 32,9 : PRINT
6030 PRINT "1 CHECK OUT A BOOK"
6040 PRINT
6050 PRINT "2 RETURN A BOOK"
6060 PRINT
6070 PRINT "3 SEE WHICH BOOKS A"
6080 PRINT " STUDENT HAS BORROWED"
6090 PRINT : PRINT : PRINT
6100 INPUT "WHICH WOULD YOU LIKE?"; W
6105 TEXT : PRINT : HOME
6110 ON W GOTO 340, 560, 850
```

INSTRUCTIONAL OBJECTIVE

To change text windows by using poke commands.

PROCEDURES

If you fill the screen before using the pokes to change the window, students can visualize the differences more easily. Lines 20 through 40 fill the screen with dots.

```
10 HOME
20 FOR I = 1 TO 960
30 ?".";
40 NEXT I
```

The POKE 32, ☐ command changes the left hand corner of the screen. POKE 33, ☐ will determine the width of the text window.

For vertical top and bottom limits use POKE 34, ☐ and POKE 35, ☐ .

POKE 32, R (R has a minimum of 0, normal is 0, maximum 39)
POKE 33, R (R has a minimum of 0, normal is 40, maximum 40)
POKE 34, R (R has a minimum of 0, normal is 0, maximum 24)
POKE 35, R (R has a minimum of 0, normal is 24, maximum 24)

```
50 POKE 32,4 : POKE 33,15
60 POKE 34,6 : POKE 35,11
70 HOME
80 ?"********"
90 ?"HHHHHHH"
100 POKE 32,15 : POKE 33,10
110 POKE 34,10 : POKE 35,13
115 HOME
120 ?"HERE I AM"
130
```

↓

Change POKEs and PRINTs to see effects.

Omit the HOME command and RUN. Another way of getting the cursor to the right position is via VTAB and use PRINT" " before line 80.

By using a long thin text window and an endless loop, create a simulation by printing :

```
100 ?"*      "
110 ?" *     "
120 ?"  *    "
130 ?"   *   "
140 ?"    *  "
150 ?"   *   "
160 ?"  *    "
170 ?"*      "
180 ?"       "
```

Ask the class why movement seems to change speed. Or use a long short window and move a character across the screen. ☐→

ADDITIONAL PROGRAMS

```
]LIST

1  HOME
2  FOR I = 1 TO 960: PRINT ".";
3  NEXT I
10  POKE 32,4: POKE 33,16
20  POKE 34,4: POKE 35,7
32  HOME
40  PRINT "***************"
50  PRINT "###############"
56  POKE 32,1: POKE 33,11
57  POKE 34,14: POKE 35,16
58  HOME
60  PRINT "HERE I AM."
100  POKE 32,25: POKE 33,12
110  POKE 34,5: POKE 35,19
115  HOME
117  SPEED= 180
119  FOR X = 1 TO 5
130  PRINT "           *"
140  PRINT "          *"
150  PRINT "         *"
160  PRINT "        *"
165  SPEED= 150
170  PRINT "       *"
171  PRINT "      *"
172  PRINT "     *"
173  PRINT "    *"
174  PRINT "   *"
180  PRINT "  *"
190  PRINT " "
200  PRINT "  *"
210  PRINT "   *"
220  PRINT "    *"
230  PRINT "     *"
240  PRINT "      *"
250  PRINT "       *"
260  PRINT "        *"
270  PRINT "         *"
280  PRINT "          *"
290  PRINT "           *"
300  NEXT
310  SPEED= 255
320  TEXT
```

```
]LIST

10  TEXT
20  HOME
30  SPEED= 255
40  FOR A = 1 TO 960
50  PRINT ".";
60  NEXT
70  FOR A = 1 TO 9
80  POKE 32,5: POKE 33,10: POKE 3
     4,5: POKE 35,7
90  HOME
100  HTAB (A)
101  PRINT "]";
102  HTAB (10 - A)
103  PRINT  CHR$ (91)
110  POKE 32,25: POKE 33,10: POKE
     34,14: POKE 35,16
120  HOME
130  HTAB (10 - A): PRINT  CHR$ (
     91);: HTAB (A): PRINT "]";
140  NEXT
150  GOTO 70
```

BEST COPY AVAILABLE

INSTRUCTIONAL OBJECTIVE

To write programs to increase speed in drill and practice routines by using a time counter.

PROCEDURES

The usual initial response from students when asked "How can we allow a limited time for an input?" is quick but wrong. They think of using a FOR--NEXT loop to introduce a delay. Let them write a program using their easy solution and see what happens. Most programs mirror one of the two examples listed below.

#1

```
10?"HOW MANY CENTIMETERS IN A METER?"
20 INPUT A
30 FOR X = 1 TO 2000 : NEXT X : ?"SORRY, TOO LATE!"
   : END
40 IF A = 100 THEN ?"GREAT!"
```

#2

```
10 ?"HOW MANY FEET IN A YARD?"
20 FOR X = 1 TO 2000
30 INPUT A
40 NEXT X
50 ?"SORRY TOO LATE!" (or IF X = 200 THEN ? "SORRY TOO
                                        LATE")
60 IF X< 2000 THEN ?"GREAT!"
```

After running their programs they can usually appreciate the magnitude of the problem.

Here is a simplistic solution:

```
10 A$ = " "
20 ?"THIS IS A QUIZ"
30 ?"4 X 3 = "
100 FOR I = 1 TO 1000
110 Z = PEEK(-16384) : POKE -16368,0
120 IF Z < 128 THEN GOTO 200
130 NEXT I
140?"SORRY, TOO LATE "; END
```

reads keyboard

resets keyboard so next character may be read in

Z = ASCII of key pressed

return key signaling end of response

```
200 IF (Z-128) = (13) THEN GOTO 300
210 A$ = A$ + CHR$(Z - 128)
220 GOTO 100
300 A = VAL A$
310 IF A = 12 THEN ?"RIGHT!" : END
320 ?"SORRY, YOU'RE WRONG.  THE CORRECT ANSWER IS 12"
```

The next program is a little more interesting in that it uses a random generator.

Change or add the following lines:

```
 30 X = INT(5*RND (1) + 1)    --random numbers from 1 - 5.
 40 Y = INT(5*RND (1) + 1)
 50 HTAB7 : ? "X"  * "Y" = ";

305 ?A
320 ?"THE CORPECT ANSWER IS " X * Y
```

INSTRUCTIONAL OBJECTIVE

To apply formulas used with second degree equations to computer programs.

PROCEDURES

(These programs assume knowledge of Algebra II).

The first program in this section will find the zero's of a quadratic and identify them as real or imaginary.

```
10 PRINT "A QUADRATIC EQUATION CAN BE WRITTEN AS AX^2+BX+C = F(X)"
20 PRINT "THE VALUE OF A CANNOT BE ZERO OR OUR EQUATION WOULD BE
   BX + C = F(X) WHICH IS LINEAR"
30 PRINT "I WILL ASK YOU FOR THREE NUMBERS, REPRESENTING THE A, B, AND
   C RESPECTIVELY"
40 INPUT "A WILL BE "; A
50 IF A = 0 THEN PRINT "A SHOULDN'T BE ZERO.  TRY AGAIN" : GOTO 20
60 INPUT "B WILL HAVE A VALUE OF  "; B
70 INPUT "C IS ";C
80 S = B^2 - 4 * A * C
90 IF S<0 THEN S = ABS (S) : GOTO 200
100 X1 = ( - B + SQR (S)) /  * A
110 X2 = ( - B - SQR (S)) /  * A
120 PRINT "THE ROOTS ARE "; X ; "AND "; X2
130 END
200 PRINT "THE ROOTS ARE IMAGINARY.  THEY ARE "; ( - B / 2 * A);
   "+"; (S / 2 * A); "I AND "; ( - B / 2 * A);" - "; (S / 2 * A);" I"
```

Another point of interest when working with quadratics is the turning point (maximum or minimum values of F(X) occur at this point.) Running through this point is either a horizontal or vertical line cutting the graph into two equal halves, that is the line of symmetry. We will find both in this program. Explain to the class that we will work with real functions only. IF $B^2 - 4AC$ is negative, we will start again.

```
10 PRINT "PLEASE ENTER THE A,B, AND C FOR THE QUADRATIC FUNCTION"
20 INPUT A,B,C : IF A = 0 THEN GOTO 10
30 PRINT "YOUR EQUATION IS ";A;"X^2 + ";B; "X + ";C;" = F(X)"
40 X = -B / (2*A)
50 PRINT "THE AXIS OF SYMMETRY IS X = ";X
60 Y = (4*A*C - B^2) /4 * A
70 PRINT "THE TURNING POINT IS ( ";X;",";Y;")"
```

ADDITIONAL PROGRAMS

The following are two programs written by first year students on curve sketching. Have your class improve them.

AREA UNDER A CURVE

```
1  HOME
2  PRINT
3  PRINT "       THE AREA UNDER A
     CURVE"
4  PRINT
5  FOR PAUSE = 1 TO 800: NEXT PAU
     SE
6  PRINT
7  PRINT "THIS PROGRAM WILL APPRO
     XIMATE THE AREA"
8  PRINT
9  PRINT "UNDER A CURVE BY FINDIN
     G THE  TOTAL"
10  PRINT
11  PRINT "AREA OF A DESIGNATED N
     UMBER OF INSCRIBED"
12  PRINT : PRINT "RECTANGLES.";
13  PRINT "PLEASE ENTER 2 NUMBERS
     "
14  PRINT
15  INPUT "FOR THE DOMAIN (BETWEE
     N .5 AND 3.5)";C,D
16  IF C < .5 OR C > 3.5 OR D < .
     5 OR D > 3.5 THEN  GOTO 13
17  PRINT : PRINT "NOW ENTER THE
     NUMBER OF RECTANGLES."
18  PRINT
19  INPUT "YOU WISH TO INSCRIBE "
     ;E
20  HCOLOR= 2
26 A = 0
28  HGR
29  HCOLOR= 2
30  HPLOT 0,80 TO 279,80
40  HPLOT 140,0 TO 140,159
50  HCOLOR= 3
60 T = 0
70  FOR X = C TO D STEP .25
80 Y = 1 / X
90  PRINT X,Y
100 L = 140 + 35 * X
110 M = 80 - 35 * Y
120  IF T = 0 THEN  HPLOT L,M
130 T = T + 1
140  HPLOT  TO L,M
150  NEXT X
160 F = (D - C) / E
170 Q = 0
180  FOR X = C TO D STEP F
190 Y = 1 / X
200 L = 140 + 35 * X
210 M = 80 - 35 * Y
220 H = 140 + 35 * (X - F)
230  HPLOT L,80 TO L,M
240  IF Q > 0 THEN  HPLOT L,M TO
     H,M
241  FOR W = 0 TO F STEP .1
242 R = 140 + 35 * (X - F + W)
245  IF Q > 10 * F THEN  HPLOT R,
     80 TO R,M
250 Q = Q + 1
251  NEXT W
260  NEXT X
263  IF X = D + F THEN  GOTO 286
265  HPLOT 140 + 35 * D,80 TO 140
      + 35 * D,M
270  FOR P = D - F TO D STEP .1
271 L = 140 + 35 * P
272 Y = 1 / D
275 M = 80 - 35 * Y
280  HPLOT L,80 TO L,M
285  NEXT P
286  PRINT
288 F = (D - C) / E
299 A = 0
300 T = 0
310  FOR X = C + F TO D STEP F
320 T = T + 1
325 Y = 1 / X
330 N = F * Y
340 A = A + N
350  NEXT X
355  IF T = E - 1 THEN  GOTO 400
360  PRINT "    THE TOTAL AREA IS
      ";A
380  GOTO 500
400 P = D
410 Y = 1 / P
420 N = F * Y
430 A = A + N
440  PRINT "   THE TOTAL AREA IS
     ";A
500  FOR PAUSE = 1 TO 2000: NEXT
     PAUSE
501  HGR
510  HOME
520  TEXT
```

BEST COPY AVAILABLE

CURVE SKETCHIN

```
1  HOME
10  PRINT "          GREETING'S AND
     SALUTATIONS!"
13  PRINT
15  PRINT "PLEASE BE SEATED, AND
     PREPARE TO BE AMAZED!"
17  PRINT
20  PRINT "THIS PROGRAM IS DESIGN
     ED TO GRAPH THE   CUBIC, QUA
     DRATIC, OR LINEAR EQUATION O
     F YOUR DREAMS!"
23  PRINT
25  PRINT "ARE YOU READY? (YES/NO
     )";
30  INPUT Q$
35  IF Q$ = "YES" THEN : GOTO 45
36  PRINT
37  PRINT "ARE YOU READY YET? (YE
     S/NO)";
40  GOTO 30
45  PRINT
46  PRINT "THE EQUATION TO BE GRA
     PHED IS IN THE    FORM OF:"
48  PRINT
50  PRINT "  Y = A*X^3 + B*X^2 +
     C*X + D"
52  PRINT
55  PRINT "WHEN YOU SEE THE BLINK
     ING SQUARE, FILL  IN THE A,B
     ,C, AND D.  A (0) IN ANY OF
       THE INPUT WILL ELIMINATE T
     HAT TERM."
57  PRINT
60  PRINT "OKAY, GO AHEAD"
61  INPUT A,B,C,D
62  PRINT
65  PRINT "YOUR EQUATION IS:"
67  PRINT
70  PRINT "   Y = ";A;"X^3 + ";B;
     "X^2 + ";C;"X + ";D
72  PRINT
75  PRINT "ARE YOU SATISFIED? (YE
     S/NO)"
77  INPUT D$
80  IF D$ = "YES" THEN : GOTO 100

82  PRINT
85  PRINT "OKAY, HERE'S YOUR SECO
     ND CHANCE"
90  GOTO 61
98  PRINT
99  PRINT
100  PRINT "AND HERE'S YOUR GRAPH
     !"
101  PRINT
105  FOR PAUSE = 1 TO 1500: NEXT
     PAUSE
1000  HGR
1400  HCOLOR= 3
1600  HPLOT 0,80 TO 279,80
1800  HPLOT 140,0 TO 140,159
1900 T = 0
2000  FOR X =  - 8 TO 8 STEP .5
2100 Y = A * X ^ 3 + B * X ^ 2 +
     C * X + D
2105 X =  INT (10 * X) / 10
2110 Y =  INT (10 * Y) / 10
2200  PRINT X,Y
2400 L = 140 + (5 * X)
2600 M = 80 - (5 * Y)
2700  IF L < 0 OR L > 279 THEN : GOTO
     3300
2750  IF M < 0 OR M > 159 THEN : GOTO
     3300
2800  IF T = 1 THEN : GOTO 3200
2900  HPLOT L,M
3000 T = T + 1
3100  GOTO 3300
3200  HPLOT  TO L,M
3300  NEXT X
3500  TEXT
3600  HOME
3700  PRINT "WOULD YOU LIKE TO TR
     Y ANOTHER GRAPH?    (YES/NO)
     "
3800  INPUT L$
3900  IF L$ = "YES" THEN : GOTO 6
     0
4000  PRINT
4100  PRINT "WELL EXCUUUUUUSE ME!
     ......AND HAVE A   GOOD DAY
     "
4200  PRINT
4250  INVERSE
4300  PRINT "                  BYE-
     BYE                "
4350  NORMAL
4400  END
```

BEST COPY AVAILABLE

INSTRUCTIONAL OBJECTIVE

To show different rates of movement across a graphics screen as a prelude to animation.

To realize that smooth animation requires more advanced techniques.

PROCEDURES

This program shows a square moving across the screen at various rates.

```
10 GR
20 COLOR = 2
30 C = INT(5 * RND (1))
40 IF C = 0 THEN P = 1
50 IF C = 1 THEN P = 100
60 IF C = 2 THEN P = 10
70 IF C = 3 then P = 50
80 IF C = 4 THEN P = 25
90 FOR X = 1 TO 34
100 HLIN X + 2, X + 4 at 8
110 FOR PAUSE = 1 TO P : NEXT PAUSE
120 COLOR = 0
130 HLIN X + 2, X + 4 at 8
140 FOR PAUSE = 1 TO P : NEXT PAUSE
150 NEXT X
160 GOTO 30
```

You could also try on C GOTO 40, 50, 60, 70, 80 -- with GOTO 90 at the end of each or ON C GOSUB 200, 300. 400, 500, 600

```
199 END
200 P = 1 : RETURN
200 P = 100 : RETURN
400 P = 10 : RETURN
500 P = 50 : RETURN
600 P = 25 : RETURN
```

INSTRUCTIONAL OBJECTIVE

To explore the world of low-resolution graphics.

PROCEDURES

This graphics program moves a figure to the end of the screen, then back again.

MOVING HORSE

```
 20 GR
 40 BODY = 7
 50 FEET = 9
 60 FACE = 15
 80 X = 10
 90 Y = 10
100 X = X + 1
101 Y = Y + 1
103 GOTO 1010
110 X = X - 1
111 Y = Y - 1
1010 COLOR = BODY
1015  IF X< 35 OR Y> 2 THEN : T = 1 : GOTO 110
1016 IF X< 2 OR Y< 2 THEN T = 0 : GOTO 100
1020 PLOT X,Y + 1
1030 HLIN X,X + 2 AT Y
1040 COLOR = FEET
1050 PLOT X, Y + 1
1060 PLOT X + 2, Y + 1
1070 COLOR = FACE
1080 PLOT X - 1, Y -1
2000 COLOR = 0
2010 PLOT X, Y -1
2020 HLIN X,X + 2 AT Y
2030 PLOT X, Y + 1
2040 PLOT X + 2, Y + 1
2050 PLOT X - 1, Y - 1
3000 IF T = 1 THEN GOTO 110
3001 IF T = 0 THEN GOTO 100
```

BEST COPY AVAILABLE

MOVING X

```
10 GR
15 X = 2
20 COLOR = 7
30 PLOT X,7
40 COLOR = 3
50 HLIN X - 2, X + 2 AT 8
60 PLOT X, 9
70 PLOT X, 10
80 COLOR = 0
90 PLOT X,7
100 HLIN X -2, X +2 AT 8
110 PLOT X, 9
120 PLOT X, 10
125 X = X + 1
127 IF X = 37 THEN X = 2
130 GOTO 20
```

THE MARTIAN

```
 1 X = 1
 10  GR
 12  COLOR= 12
 14  GOSUB 900
 16  HLIN 12,23 AT 20
 18  GOSUB 900
 20  HLIN 11,24 AT 21
 22  GOSUB 900
 24  HLIN 11,12 AT 22
 26  GOSUB 900
 27  GOSUB 2000
 28  HLIN 16,19 AT 22
 30  GOSUB 900
 32  HLIN 23,24 AT 22
 34  GOSUB 900
 36  HLIN 10,12 AT 23
 37  GOSUB 2000
 38  GOSUB 900
 40  HLIN 16,19 AT 23
 42  GOSUB 900
 44  HLIN 23,25 AT 23
 45  GOSUB 2000
 46  GOSUB 900
 48  HLIN 10,12 AT 24
 50  GOSUB 900
 52  HLIN 16,19 AT 24
 53  GOSUB 2000
 54  GOSUB 900
 56  HLIN 23,25 AT 24
 58  GOSUB 900
 60  HLIN 10,25 AT 25
 62  GOSUB 900
 64  HLIN 10,25 AT 26
 66  GOSUB 900
 68  HLIN 10,25 AT 27
 69  GOSUB 2000
 70  GOSUB 900
 72  HLIN 10,25 AT 28
 74  GOSUB 900
 76  HLIN 10,25 AT 29
 77  GOSUB 2000
 78  GOSUB 900
 80  HLIN 10,25 AT 30
 82  GOSUB 900
 84  HLIN 10,17 AT 31
 85  GOSUB 2000
 86  GOSUB 900
 88  HLIN 21,25 AT 31
 90  GOSUB 900
 92  HLIN 10,14 AT 32
 94  GOSUB 900
 96  HLIN 18,19 AT 32
 97  GOSUB 2000
 98  GOSUB 900
 100  HLIN 23,25 AT 32
 102  GOSUB 900
 104  HLIN 11,12 AT 33
 106  GOSUB 900
 108  HLIN 16,24 AT 33
 110  GOSUB 900
```

BEST COPY AVAILABLE

THE MARTIAN (con't.)

```
112  HLIN 11,24 AT 34
114  GOSUB 900
116  HLIN 12,23 AT 35
118  GOSUB 900
120  COLOR= 11
121  GOSUB 2000
122  HLIN 13,15 AT 33
124  GOSUB 900
126  HLIN 15,17 AT 32
128  GOSUB 900
130  HLIN 17,20 AT 31
131  GOSUB 900
132  HLIN 20,22 AT 32
133  GOSUB 900
134  COLOR= 15
135  GOSUB 2000
136  HLIN 13,15 AT 22
138  HLIN 13,15 AT 23
140  HLIN 13,14 AT 24
142  HLIN 21,22 AT 22
144  HLIN 20,22 AT 23
146  HLIN 20,22 AT 24
148  GOSUB 900
150  COLOR= 13
151  GOSUB 2000
152  VLIN 17,19 AT 13
154  VLIN 17,19 AT 22
156  GOSUB 900
158  VLIN 15,17 AT 12
160  VLIN 15,17 AT 23
162  GOSUB 900
164  VLIN 13,15 AT 11
166  VLIN 13,15 AT 24
168  GOSUB 900
170  VLIN 10,13 AT 10
172  VLIN 10,13 AT 25
174  GOSUB 900
177 M = 0
179  GOSUB 2000
180  COLOR= 14
185  GOSUB 600
187 M = M + 1
190  IF M = 17 THEN : GOTO 245
192  GOSUB 900
195  COLOR= 9
200  GOSUB 600
205 M = M + 1
207  GOSUB 900
210  COLOR= 4
215  GOSUB 600
220 M = M + 1
222  GOSUB 900
225  COLOR= 3
230  GOSUB 600
235 M = M + 1
237  GOSUB 900
240  GOTO 180
243  PRINT
246  PRINT
248  PRINT ""
249  PRINT  TAB( 18)"HI!!"
250  PRINT "              THE MARTI
     AN"
270  END
600  HLIN 7,9 AT 8
610  HLIN 7,9 AT 9
620  HLIN 7,9 AT 10
630  HLIN 26,28 AT 8
640  HLIN 26,28 AT 9
650  HLIN 26,28 AT 10
660  RETURN
900  FOR PAUSE = 1 TO 200: NEXT P
     AUSE
910  RETURN
2000 A =  INT (10 *  RND (1)) + 1

2005 B =  INT (5 *  RND (1))
2010 C = A + B
2015  PRINT  TAB( 15)X;")  ";A" +
      ";B;" = ";
2020  INPUT R
2030 X = X + 1
2040  INVERSE
2045  PRINT
2050  IF R <  > C THEN : PRINT  TAB(
     15)A;" + ";B;"= ";C;"
        "
2055  NORMAL
2056  PRINT
2060  IF R <  > C THEN  GOTO 2000

2070  RETURN
```

BEST COPY AVAILABLE